Казанский Государственный Университет

А.В. Казанцев

Основы компьютерной графики

Тексты специального курса лекций

Часть 1

Математический аппарат компьютерной графики

Казань 2001, v. 1.0

ОГЛАВЛЕНИЕ

Предисловие

Тексты лекций представляют собой учебное пособие для начинающих осваивать компьютерную графику. Они написаны на основе специального курса лекций, читаемых автором в течение четырех лет в Казанском государственном университете на факультете вычислительной математики и кибернетики. Здесь содержится информация, необходимая при разработке трехмерных приложений компьютерной графики.

Многие из книг по компьютерной графике глубоко исследуют узкоспециализированные области, такие как разработка библиотек подпрограмм для реализации метода обратного хода лучей или скоростных методов изображения трехмерных сцен которые используются в компьютерных играх, либо низкоуровневому программированию видеоадаптеров. При этом, например, для студентов, только начинающих вникать в эту область, часто недостает информации общеознакомительного плана, позволяющей сориентироваться в стремительно расширяющейся области компьютерной графики. Данный материал призван хотя бы отчасти восполнить указанный пробел.

Рассмотрим кратко начало развития компьютерной графики. Отправной точкой можно считать 1930 год, когда в США нашим соотечественником Владимиром Зворыкиным, работавшим в компании “Вестингхаус” (Westinghouse), была изобретена электронно-лучевая трубка (ЭЛТ), впервые позволяющая получать изображения на экране без использования механических движущихся частей. Именно ЭЛТ является прообразом современных телевизионных кинескопов и компьютерных мониторов. Началом эры собственно компьютерной графики можно считать декабрь 1951 года, когда в Массачусеттском технологическом институте (МТИ) для системы противовоздушной обороны военно-морского флота США был разработан первый дисплей для компьютера “Вихрь”. Изобретателем этого дисплея был инженер из МТИ Джей Форрестер.

Одним из отцов-основателей компьютерной графики считается Айвен Сазерленд (Ivan Sotherland), который в 1962 году все в том же МТИ создал программу компьютерной графики под названием “Блокнот” (Sketchpad). Эта программа могла рисовать достаточно простые фигуры (точки, прямые, дуги окружностей), могла вращать фигуры на экране. После этой программы некоторые крупные фирмы, такие как “Дженерал моторз”, “Дженерал электрик”, приступили к разработкам в области компьютерной графики. В

1965 году фирма IBM выпустила первый коммерческий графический терминал под названием IBM-2250. В конце 70-х годов для космических кораблей “Шаттл” появились летные тренажеры, основанные на компьютерной графике. В 1982 году на экраны кинотеатров вышел фильм “Трон”в котором впервые использовались кадры, синтезированные на компьютере. Еще в 1979 году Джордж Лукас, глава фирмы “Lucasfilm” и создатель сериала “Звездные войны”, организовал в своей фирме отдел, который занимался внедрением последних достижений компьютерной графики в кинопроизводство.

Существуют фирмы, специализирующиеся на разработке компьютеров для графических приложений, такие как “Silicon Graphics”, “Evans&Sotherland”. Области приложения компьютерной графики в настоящее время очень широки. В промышленности используется компьютерное моделирование процессов с графическим отображением происходящего на экране. Разработка новых автомобилей проходит на компьютере от стадии первичных эскизов внешнего вида корпуса автомобиля до рассмотрения поведения деталей автомобиля в различных дорожных условиях. В медицине применяются компьютерные томографы, позволяющие заглянуть внутрь тела и поставить правильный диагноз. В архитектуре широко применяются системы автоматизированного проектирования (CAD – Computer Aided Design) которые позволяют разработать полный проект здания, основываясь на методах компьютерной графики. Химики изучают сложные молекулы белков пользуясь средствами компьютерного отображения данных. В телевидении и кинематографии компьютерная графика стала обыденным явлением. В мире регулярно проводятся выставки, например, такие как SIGGRAPH, картин нарисованных с помощью компьютера. В математике развитие фракталов было бы невозможно без компьютеров с соответствующими средствами графического отображения данных. Средства мультимедиа привели к появлению новых источников информации объединяющих в себе статические и видео изображения, текст и звук. Новейшие операционные системы работают в графическом режиме и изначально реализуют в своих функциях методы компьютерной графики.

Элементы аналитической геометрии

Для того чтобы уметь синтезировать изображения на экране компьютера необходимо предложить способ математического описания объектов в трехмерном пространстве или на плоскости. Окружающий нас мир с точки зрения практических приложений описывают как трехмерное евклидово пространство. Под описанием трехмерного объекта будем

понимать знание о положении каждой точки объекта в пространстве в любой момент времени. Положение точек в пространстве удобно описывается с помощью декартовой системы координат.

Для того чтобы ввести декартову систему координат проведем три направленные прямые линии, не лежащие в одной плоскости, которые называются осями, в трехмерном пространстве так чтобы они пересекались в одной точке – начале координат. Выберем на этих осях единицу измерения. Тогда положение любой точки в пространстве будем описывать через координаты этой точки, которые представляют собой расстояния от начала координат до проекций точки на соответствующие оси координат. Проекцией точки на координатную ось называется точка пересечения плоскости, проходящей через заданную точку и параллельной плоскости, образованной двумя другими осями координат. Например, на рис. 1 проекцией точки P на ось Ox является точка Q, которая принадлежит плоскости, параллельной плоскости zOy.

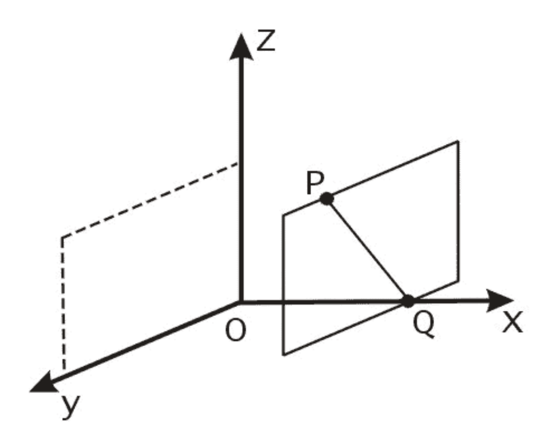


Рис. 1. Нахождение координаты $x = Q$ точки P.

В общем случае оси системы координат могут располагаться под произвольными, хотя и фиксированными углами друг относительно друга. Для практических расчетов гораздо удобнее когда эти оси расположены взаимно перпендикулярно. Такая система координат называется ортогональной. В ортогональной системе координат проекцией точки P на ось является единственная точка на оси такая, что отрезок прямой, проведенной из этой точки к точке P является перпендикулярным к данной оси.

Таким образом, положение в пространстве точки P описывается ее координатами, что записывается как $P = (x, y, z)$. Взаимное расположение

осей в ортогональной системе координат в трехмерном пространстве может быть двух видов. Проведем ось Ox слева направо, а ось Oy снизу вверх, как показано на рис. 2.

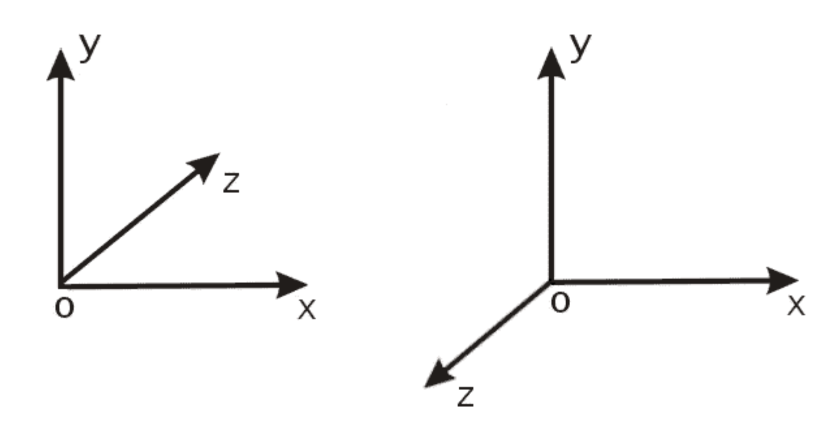


Рис.2. Левосторонняя и правосторонняя системы координат.

Ось Oz при этом может проходить как в направлении от наблюдателя в плоскость листа, так и от плоскости листа к наблюдателю. В первом случае система координат будет называться левой или левосторонней, а во втором случае – правой или правосторонней. Более точное определение правой и левой систем координат можно дать следующее. Если посмотреть из положительной полуоси Oz в направлении начала координат, то для совмещения положительной полуоси Ox с положительной полуосью Oy необходимо повернуть Ox относительно начала координат против часовой стрелки – в этом случае имеем правую систему координат; если же поворот производится по часовой стрелке – то система координат левая[*]. Существует также легкий способ определения вида системы координат по правой или левой руке, как показано на рис. 3. Для левой руки большой, указательный и средний пальцы формируют левую тройку ортогональных векторов. То же относится и к их циклическим перестановкам.

[*] В этом определении при замене, скажем, оси Oz на ось Ox остальные оси заменяются по правилу циклической перестановки, то есть Oy заменится на Oz, а Ox заменится на Oy. Всего циклических перестановок может быть три: (x,y,z)→(y,z,x)→(z,x,y).

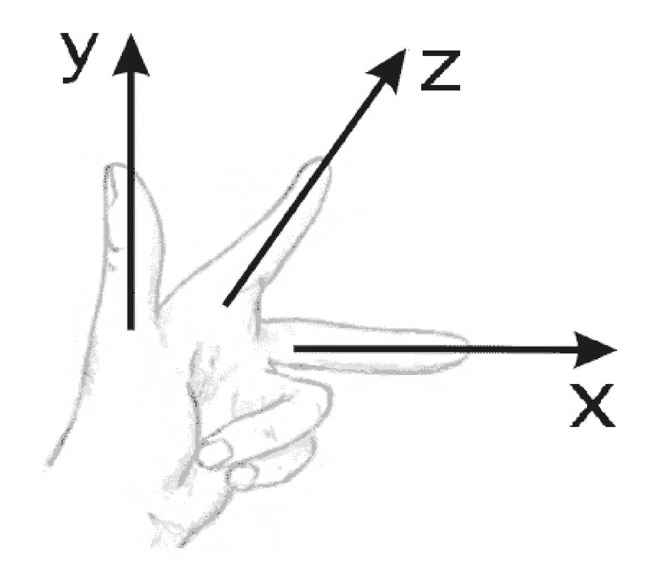


Рис. 3. Определение левосторонней системы координат по левой руке.

Декартовы координаты точек позволяют описывать статичное положение объектов в пространстве. Однако для проведения каких-либо действий над объектами необходимо иметь дополнительные математические конструкции. В качестве одной из таких конструкций применяют радиус-векторы. Радиус-векторы обладают всеми свойствами векторов, но имеют одну особенность: начало радиус-вектора находится всегда в начале координат, а конец радиус-вектора лежит в некоторой точке пространства. Это свойство радиус-векторов позволяет поставить во взаимно однозначное соответствие всем точкам пространства соответствующие им радиус-векторы. Формально это соответствие запишем в следующем виде. Пусть точка P имеет координаты (x, y, z), то есть $P = (x, y, z)$, и $\boldsymbol{p} = x\boldsymbol{i} + y\boldsymbol{j} + z\boldsymbol{k}$ – радиус-вектор, конец которого находится в точке P, где $\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}$ – тройка единичных базисных векторов, или просто нормированный базис. Тогда точке P взаимно однозначно соответствует радиус-вектор $\boldsymbol{p}$, или $P = (x, y, z) \Leftrightarrow x\boldsymbol{i} + y\boldsymbol{j} + z\boldsymbol{k} = \boldsymbol{p}$. Таким образом, можно легко переходить от координат точек к радиус-векторам и обратно. Далее мы увидим что представление радиус-вектора в виде линейной комбинации векторов базиса имеет вполне конкретное практическое применение. Отметим, что радиус-вектор иногда определяют как преобразование переноса точки из начала координат в заданную точку пространства с известными координатами. При

этом умножение радиус-вектора $\boldsymbol{p}$ на число a означает перенос точки из начала координат в направлении вектора $\boldsymbol{p}$ на расстояние $a|\boldsymbol{p}|$, где прямые скобки означают операцию взятия модуля вектора:

$$|\boldsymbol{p}| = \sqrt{x^2 + y^2 + z^2}.$$

Сложение радиус-векторов $\boldsymbol{p}+\boldsymbol{q}$ можно рассматривать как перенос точки P по направлению вектора $\boldsymbol{q}$ на расстояние $|\boldsymbol{q}|$.

Рассмотрим теперь каким образом можно использовать координаты точек и радиус-векторы для описания прямых и плоскостей в трехмерном пространстве. Под описанием прямой понимаем знание того принадлежит ли точка с заданными координатами нашей прямой или нет. То есть нужно получить некую математическую зависимость или уравнение прямой. Мы получим уравнение прямой двумя способами.

Во-первых, известно, что две различные точки определяют в пространстве прямую. Выберем в пространстве две точки $P_1 = (x_1, y_1, z_1) \Leftrightarrow \boldsymbol{p}_1$ и $P_2 = (x_2, y_2, z_2) \Leftrightarrow \boldsymbol{p}_2$ и проведем через них прямую, как показано на рис 4.

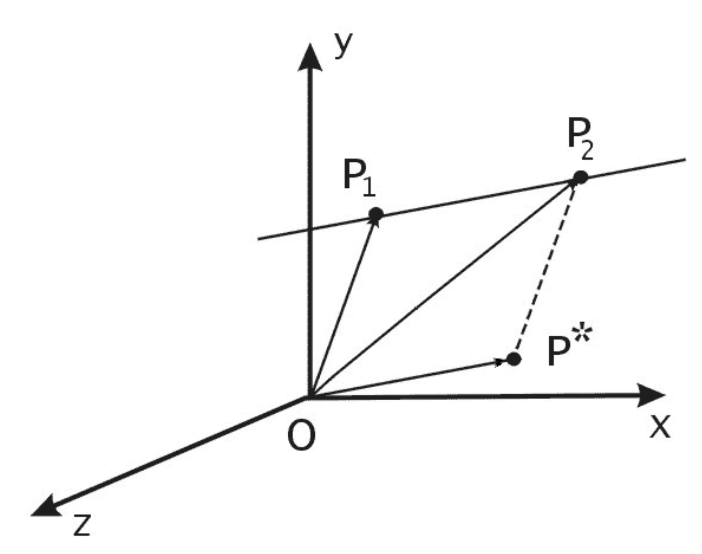


Рис. 4. Вывод уравнения прямой в трехмерном пространстве.

Проведем от точки P_1 к точке P_2 вектор $\boldsymbol{p}^* = \boldsymbol{p}_2 - \boldsymbol{p}_1$. Тогда радиус-вектор $\boldsymbol{p}$, определяющий некоторую точку на прямой, можно получить сложением, например, вектора $\boldsymbol{p}_1$ и вектора $\boldsymbol{p}^*$, умноженного на некоторое число μ. Или $\boldsymbol{p} = \boldsymbol{p}(\mu) = \boldsymbol{p}_1 + \mu\boldsymbol{p}^*$. Фактически мы уже получили уравнение прямой, но не через координаты двух точек на прямой, а другим способом, с помощью, так называемых, базового радиус-вектора $\boldsymbol{p}_1$ и направляющего

радиус-вектора $\boldsymbol{p}^*$. Преобразуем это уравнение к виду в котором используются только координаты двух исходных векторов $\boldsymbol{p}_1$ и $\boldsymbol{p}_2$:

$$\boldsymbol{p} = \boldsymbol{p}_1 + \mu\boldsymbol{p}^* = \boldsymbol{p}_1 + \mu(\boldsymbol{p}_2 - \boldsymbol{p}_1) \Rightarrow \boldsymbol{p} - \boldsymbol{p}_1 = \mu(\boldsymbol{p}_2 - \boldsymbol{p}_1) \qquad (1)$$

Из этого векторного равенства получаем три равенства для соответствующих координат:

$$\begin{cases} x - x_1 = \mu(x_2 - x_1) \\ y - y_1 = \mu(y_2 - y_1) \\ z - z_1 = \mu(z_2 - z_1) \end{cases}$$

Попарно разделив эти уравнения друг на друга для того чтобы избавится от коэффициента μ, получаем следующую систему уравнений, определяющую нашу прямую в трехмерном пространстве:

$$\begin{cases} (x - x_1)(y_2 - y_1) = (x_2 - x_1)(y - y_1) \\ (y - y_1)(z_2 - z_1) = (y_2 - y_1)(z - z_1) \\ (z - z_1)(x_2 - x_1) = (z_2 - z_1)(x - x_1) \end{cases} \qquad (2)$$

В практических задачах иногда бывает нужно узнать лежит ли некоторая точка, принадлежащая прямой, внутри отрезка, заданного координатами своих концов на данной прямой, или снаружи. Для решения этой задачи перепишем уравнение (1) в следующем виде:

$$\boldsymbol{p} = (1 - \mu)\boldsymbol{p}_1 + \mu\boldsymbol{p}_2 \qquad (3)$$

При $\mu \in [0,1]$ получаем точки прямой, лежащие между $\boldsymbol{p}_1$ и $\boldsymbol{p}_2$. При $\mu < 0$ – точки лежащие на прямой за $\boldsymbol{p}_1$, при $\mu > 1$ – точки, лежащие на прямой за $\boldsymbol{p}_2$. Для проверки этого просто подставьте в уравнение вместо μ значения 0 и 1.

Перейдем теперь к выводу уравнения плоскости. Мы сможем получить его тремя путями. Но прежде напомним определение скалярного произведения. Для двух радиус-векторов $\boldsymbol{p}$ и $\boldsymbol{q}$ скалярным произведением назовем число $\boldsymbol{p} \cdot \boldsymbol{q} = |\boldsymbol{p}||\boldsymbol{q}| Cos\alpha$, где α - угол между векторами $\boldsymbol{p}$ и $\boldsymbol{q}$. Для векторов запись вида $\boldsymbol{pq}$ или $(\boldsymbol{p},\boldsymbol{q})$ также будем считать скалярным

произведением. Скалярное произведение можно выразить через координаты векторов:

$$\boldsymbol{p}\cdot\boldsymbol{q} = (p_1\boldsymbol{i} + p_2\boldsymbol{j} + p_3\boldsymbol{k})(q_1\boldsymbol{i} + q_2\boldsymbol{j} + q_3\boldsymbol{k}) = p_1q_1 + p_2q_2 + p_3q_3 \ ,$$

так как при раскрытии скобок скалярные произведения перпендикулярных векторов базиса по определению обращаются в ноль.

Используем свойства скалярного произведения для вывода уравнения плоскости. Рассмотрим некоторую плоскость в пространстве и некоторую точку $\boldsymbol{a}$, про которую мы знаем, что она лежит в этой плоскости, как показано на рисунке 5.

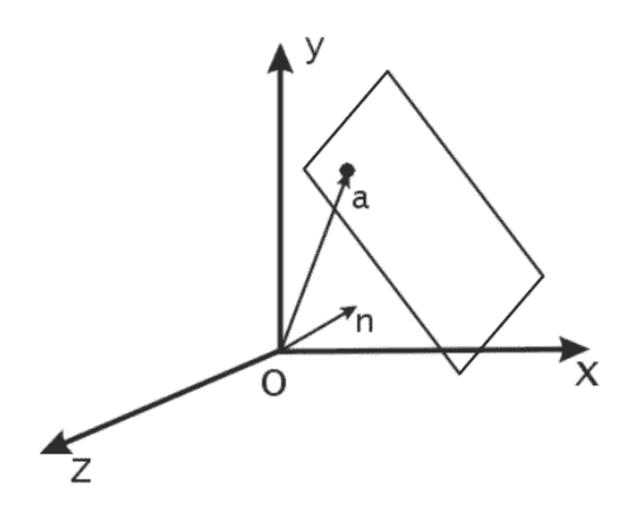


Рис. 5. Вывод уравнения плоскости в трехмерном пространстве.

Возьмем также некоторый радиус-вектор $\boldsymbol{n}$, перпендикулярный нашей плоскости. Этот вектор назовем нормалью к плоскости. Пусть теперь требуется определить принадлежит ли некоторая точка (или радиус-вектор) $\boldsymbol{p}$ плоскости или нет. Для этого заметим, что для любой точки $\boldsymbol{p}$, принадлежащей плоскости, вектор $(\boldsymbol{p}-\boldsymbol{a})$ и радиус-вектор нормали $\boldsymbol{n}$ – перпендикулярны. А это значит, что их скалярное произведение равно нулю:

$$\boldsymbol{n}(\boldsymbol{p}-\boldsymbol{a}) = 0 \qquad (4)$$

Так мы уже получили уравнение плоскости. Раскроем скобки и запишем его в более удобном виде: $\boldsymbol{np} = c$, где константа $c = \boldsymbol{na}$. Если $\boldsymbol{n} = (A, B, C)$, а $\boldsymbol{p} = (x, y, z)$, то в координатной записи наше уравнение плоскости запишется в виде

$$Ax + By + Cz = c \qquad (5)$$

Известно что плоскость может быть задана тремя точками, лишь бы они не лежали на одной прямой, то есть если они не коллинеарны. Получим уравнение плоскости для трех заданных точек. Для этого рассмотрим определение векторного произведения. Результатом векторного произведения двух векторов $\boldsymbol{p} \times \boldsymbol{q}$ является вектор $\boldsymbol{r}$, модуль которого равен $|\boldsymbol{p} \times \boldsymbol{q}| = |\boldsymbol{p}||\boldsymbol{q}| Sin\alpha$, и направлен он перпендикулярно плоскости в которой лежат векторы $\boldsymbol{p}$ и $\boldsymbol{q}$, причем векторы $\boldsymbol{p,q,r}$ – образуют правую тройку векторов (см. определение правой системы координат), здесь α также угол между векторами $\boldsymbol{p}$ и $\boldsymbol{q}$. Для векторов единичного базиса, образующих правую тройку, как следует из определения: $\boldsymbol{i} \times \boldsymbol{j} = \boldsymbol{k}$, $\boldsymbol{j} \times \boldsymbol{k} = \boldsymbol{i}$, $\boldsymbol{k} \times \boldsymbol{i} = \boldsymbol{j}$. Векторное произведение также подчиняется дистрибутивному закону как и скалярное произведение. Однако векторное произведение не коммутативно, а именно, если для векторов $\boldsymbol{u} \times \boldsymbol{v} = \boldsymbol{w}$, то $\boldsymbol{v} \times \boldsymbol{u} = -\boldsymbol{w}$, что также прямо следует из определения. Координаты векторного произведения легко получить разложив векторы, участвующие в произведении, по базису, а затем раскрыв скобки, подобно тому как это уже было проделано для скалярного произведения. Есть и другой, неформальный, но легче запоминаемый способ получения координат векторного произведения, с помощью разложения следующего определителя по его первой строке. Если $\boldsymbol{p} = (p_1, p_2, p_3)$ и $\boldsymbol{q} = (q_1, q_2, q_3)$, тогда

$$\boldsymbol{p} \times \boldsymbol{q} = \begin{vmatrix} \boldsymbol{i} & \boldsymbol{j} & \boldsymbol{k} \\ p_1 & p_2 & p_3 \\ q_1 & q_2 & q_3 \end{vmatrix} = (p_2 q_3 - p_3 q_2) \cdot \boldsymbol{i} + (p_1 q_3 - p_3 q_1) \cdot \boldsymbol{j} + (p_1 q_2 - p_2 q_1) \cdot \boldsymbol{k}$$

Сведем теперь условия в новой постановке задачи нахождения уравнения плоскости к предыдущему случаю, где мы использовали вектор

нормали. Пусть заданы фиксированные векторы $\boldsymbol{p}, \boldsymbol{q}$ и $\boldsymbol{r}$, не лежащие на одной прямой, определяющие плоскость, уравнение которой требуется получить (рис. 6).

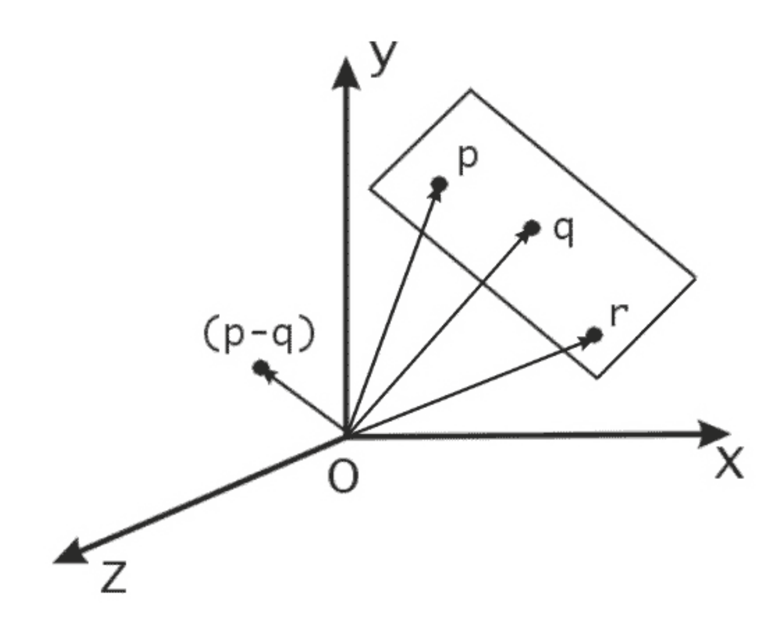


Рис. 6. Вывод уравнения плоскости проходящей через три точки.

Результат векторного произведения любых двух неколлинеарных векторов, параллельных нашей плоскости, будет вектором перпендикулярным плоскости. И как раз такими являются векторы разности $(\boldsymbol{p} - \boldsymbol{q})$ и $(\boldsymbol{p} - \boldsymbol{r})$. Выберем их векторное произведение в качестве вектора нормали, то есть $\boldsymbol{n} = (\boldsymbol{p} - \boldsymbol{q}) \times (\boldsymbol{p} - \boldsymbol{r})$. Тогда, если $\boldsymbol{x}$ – произвольный радиус-вектор, принадлежащий плоскости, то искомым уравнением плоскости будет, аналогично формуле (4):

$$[(\boldsymbol{p} - \boldsymbol{q}) \times (\boldsymbol{p} - \boldsymbol{r})] \cdot (\boldsymbol{x} - \boldsymbol{q}) = 0,$$

причем в последней скобке вместо вектора $\boldsymbol{q}$ можно было использовать, например, векторы $\boldsymbol{p}$ или $\boldsymbol{r}$. Не будем далее расписывать это уравнение через координаты, так как это не трудно проделать самостоятельно.

Рассмотрим еще несколько определений и типичных задач, решение которых не должно вызывать замешательств.

Иногда бывает необходимо вычислить длину проекции радиус-вектора не на ось системы координат, а на другой радиус-вектор. Найдем длину проекции вектора $\boldsymbol{a}$ на вектор $\boldsymbol{b}$. Эта ситуация изображена на рис.7, из которого, очевидно, следует и решение задачи.

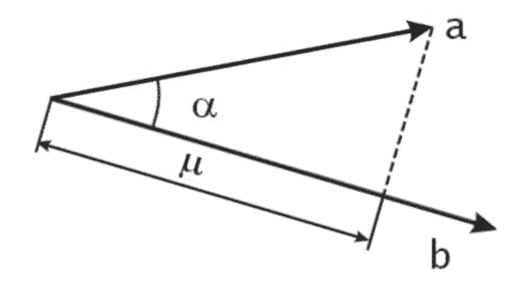


Рис. 7. Проекция вектора a на вектор b.

Искомая длина проекции: $\mu = |\boldsymbol{a}| Cos\alpha = |\boldsymbol{a}| \frac{\boldsymbol{a} \cdot \boldsymbol{b}}{|\boldsymbol{a}||\boldsymbol{b}|} = \boldsymbol{a} \cdot \frac{\boldsymbol{b}}{|\boldsymbol{b}|}$.

Как видно, если длина вектора, на который проецируется другой вектор, равна единице, то длина проекции будет просто равна скалярному произведению этих векторов.

С помощью формулы длины проекции вектора на вектор можно еще одним способом получить уравнение плоскости, если заметить, что длины проекций радиус-векторов, принадлежащих плоскости, на вектор нормали к плоскости всегда равны между собой.

Решим задачу нахождения минимального расстояния от начала координат до плоскости. Очевидно, что это расстояние необходимо откладывать вдоль прямой, определяемой вектором нормали к плоскости. Но для нахождения этого расстояния надо найти сначала точку пересечения прямой с плоскостью. Поэтому решим в общем виде задачу нахождения точки пересечения прямой и плоскости. Пусть искомая точка, или соответствующий радиус-вектор называется $\boldsymbol{x}$. Тогда эта точка должна одновременно удовлетворять уравнениям прямой и плоскости, например, $\boldsymbol{x} = \boldsymbol{p}_1 + \mu \boldsymbol{p}^*$ *и* $\boldsymbol{nx} = c$. Подставив $\boldsymbol{x}$ из первого уравнения во второе, найдем

значение константы μ, которое затем подставим в исходное уравнение прямой для получения координат искомой точки:

$$\boldsymbol{n}\left(\boldsymbol{p}_1 + \mu\boldsymbol{p}^*\right) = c \Rightarrow \quad \boldsymbol{np}_1 + \mu\boldsymbol{np}^* = c \ \Rightarrow \mu = \frac{c - \boldsymbol{np}_1}{\boldsymbol{np}^*}$$

Уравнение прямой вдоль вектора нормали к плоскости запишем как $\boldsymbol{x} = \mu \cdot \boldsymbol{p}^*$. Перед тем как подставить в это уравнение выражение для μ, заметим, что для нашей прямой, базовый вектор $\boldsymbol{p}_1$ равен нулю, а направляющий вектор совпадает с вектором нормали $\boldsymbol{p}^* = \boldsymbol{n}$. Учитывая это, запишем:

$$\boldsymbol{x} = \frac{c}{\boldsymbol{n} \cdot \boldsymbol{n}} \cdot \boldsymbol{n} = \frac{c}{|\boldsymbol{n}|^2} \cdot \boldsymbol{n}$$

Отсюда искомое расстояние от начала координат до плоскости равно

$$|\boldsymbol{x}| = \frac{c}{|\boldsymbol{n}|}$$

В том случае когда вектор нормали $\boldsymbol{n}$ является нормированным, константа c в уравнении плоскости равна расстоянию от начала координат до данной плоскости.

Кроме определения положения точек в пространстве радиус-векторы также определяют некоторое направление в пространстве. Направление, определяемое радиус-вектором, удобно описывать с помощью, так называемых, направляющих косинусов. Пусть радиус-вектор $\boldsymbol{p} = (p_1, p_2, p_3)$ составляет с осями координат Ox, Oy и Oz углы, соответственно, α_x, α_y и α_z (рисунок 8). Тогда его направляющими косинусами будут:

$$Cos\,\alpha_x = \frac{p_1}{|\mathbf{p}|},\ Cos\,\alpha_y = \frac{p_2}{|\mathbf{p}|},\ Cos\,\alpha_z = \frac{p_3}{|\mathbf{p}|}$$

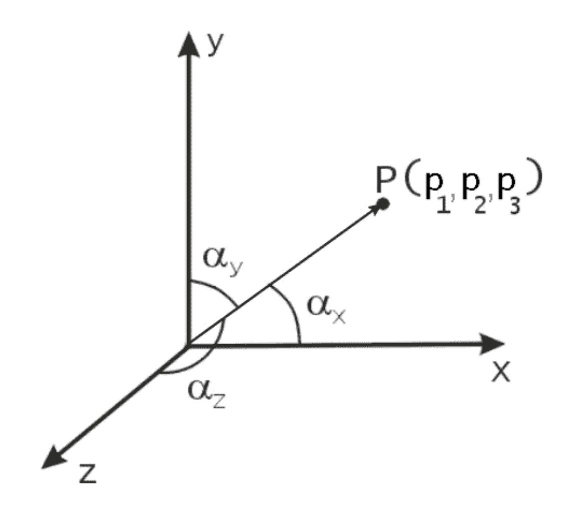


Рис. 8. Направляющие косинусы.

Отсюда, очевидно, вытекают следующие свойства направляющих косинусов:

$$Cos^2\alpha_x + Cos^2\alpha_y + Cos^2\alpha_z = 1.$$

Направляющие косинусы пропорциональны соответствующим координатам:

$$Cos\alpha_x : Cos\alpha_y : Cos\alpha_z = p_1 : p_2 : p_3 \ ,$$

а в случае, когда вектор $\boldsymbol{p}$ нормирован, значения его координат равны соответствующим направляющим косинусам.

Рассмотрим далее функциональное представление плоскости. Для этого в уравнении (5) перенесем константу из правой части в левую и запишем функцию трех переменных $f(x, y, z) = Ax + By + Cz - c$. Если подставить координаты точки, принадлежащей данной плоскости в это уравнение, то $f(x, y, z) = 0$. Если же точка не принадлежит плоскости, то значение функции, очевидно, будет больше или меньше нуля. Интересен тот факт, что для точек, лежащих по одну и ту же сторону от плоскости функция $f(x, y, z)$ имеет всегда один и тот же знак. С помощью чертежа на рисунке 9 , легко показать, что для точек лежащих в полупространстве, порождаемом плоскостью и содержащем начало координат, функция $f(x, y, z)$ отрицательна, а для точек лежащих в другом полупространстве, как, например, для точки $\boldsymbol{a}$ на рисунке, она положительна. В общем же случае необходимо учитывать направление вектора нормали.

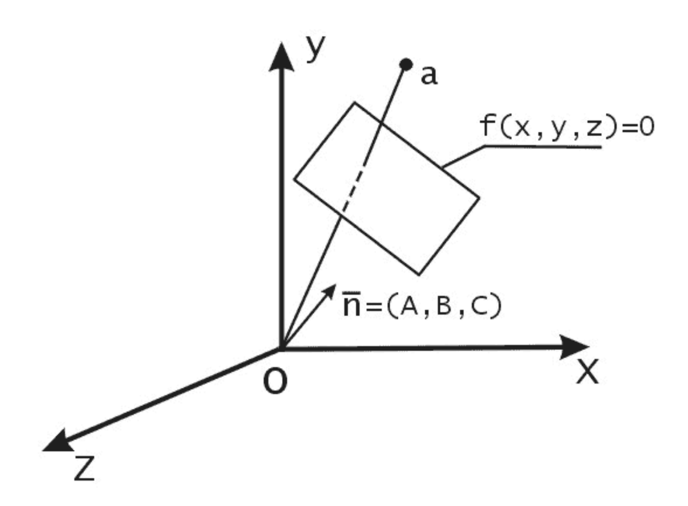


Рис. 9. Функциональное представление плоскости.

Свойство сохранения знака функции $f(x, y, z)$ удобно использовать в алгоритмах удаления невидимых ребер и граней для определения того лежат ли точки по одну сторону от плоской грани или нет. Для этого достаточно лишь подставить значения координат точек в функциональное представление плоскости, определяемой соответствующей гранью и проверить совпадают знаки функции или нет. Аналогичные рассуждения можно проделать и для более простого случая прямой на плоскости. Тогда для любой точки на плоскости можно определить ее нахождение в одной их полуплоскостей на которые прямая делит плоскость. Это свойство используется в следующем примере.

Рассмотрим далее три метода решения классической задачи определения принадлежности точки внутренней или граничной области треугольника. Эта задача имеет, конечно же, много решений, некоторые из которых может придумать и сам читатель. Пусть на плоскости xOy заданы три точки $A = (A_x, A_y), B = (B_x, B_y)$ и $C = (C_x, C_y)$, образующие треугольник (рис. 10).

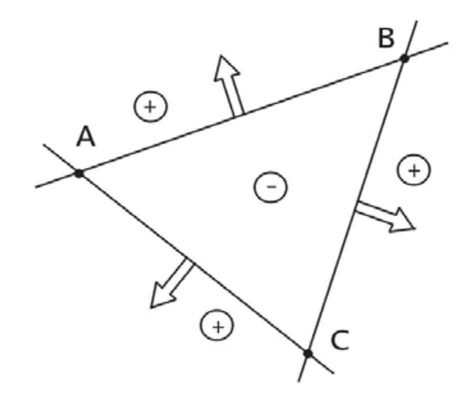


Рис. 10. Внутренняя область треугольника соответствует отрицательным направлениям векторов нормалей.

Через каждую пару вершин треугольника можно провести прямую. Замкнутая область пересечения трех полуплоскостей, образованных этими прямыми есть внутренняя область треугольника. Пользуясь вектором нормали $\boldsymbol{n} = (L, M)$ можно записать уравнение прямой на плоскости: $Lx + My + N = 0$. Идея первого метода состоит в том, чтобы записать функциональные представления уравнений прямых, образующих стороны треугольника, таким образом, чтобы внутренняя область треугольника соответствовала, например, отрицательным значениям. Тогда условием принадлежности внутренней области треугольника будут отрицательные значения трех функциональных уравнений прямых при подстановке координат проверяемой точки. Основной проблемой в этом методе является правильный выбор направления вектора нормали к прямой.

Следующий метод основан на преобразовании треугольника с помощью операции переноса таким образом чтобы проверяемая точка совпала с началом координат. Поворотом плоскости вокруг начала координат расположим одну (любую) из вершин треугольника на оси Oy. Тогда если знаки координат x оставшихся двух точек совпадают, то искомая точка лежит вне треугольника. Если же знаки различны, то берем следующую из оставшихся вершин треугольника и поворотом плоскости устанавливаем ее на ось Oy. После чего вновь проверяем знаки координат x двух других вершин, и т.д.

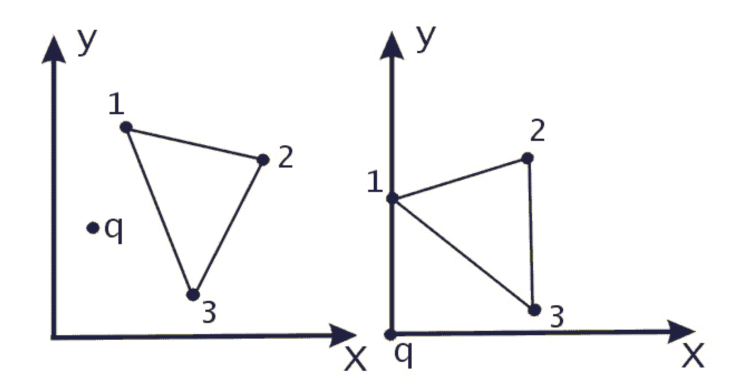


Рис. 11. Точка q вне треугольника.

Условием принадлежности точки внутренней области треугольника будет несовпадение знаков x - координат оставшихся двух вершин после каждого из трех поворотов.

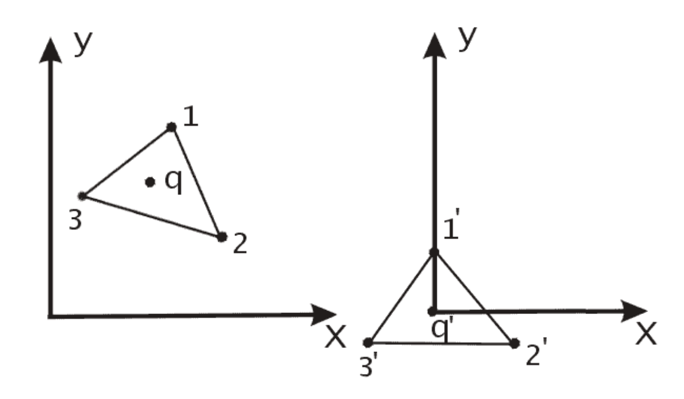


Рис. 12.Точка q внутри треугольника.

Нахождение точки на одной из сторон треугольника легко определяется по несовпадению знаков y-координат двух вершин которые после одного из поворотов оказались лежащими на оси Oy. Этот метод эффективен когда больше вероятность что точка лежит вне треугольника. Отрицательной его чертой является необходимость вычисления синусов и косинусов углов при повороте системы координат.

Третий из приводимых здесь методов представляется наиболее компактным и скоростным с вычислительной точки зрения. Этот метод был предложен автору Д. Чистяковым в 1999 году. Заметим, что очень просто можно определить принадлежность точки внутренней области треугольника – единичного симплекса, то есть треугольника, образованного точками с координатами $P=(0,0)$, $Q=(1,0)$, $R=(0,1)$. Для этого достаточно чтобы координаты искомой точки имели значения в отрезке $(0,1)$ и выполнялось условие $x+y<1$, где x и y - координаты точки. Заметим также, что с помощью аффинных преобразований на плоскости или непрерывных деформаций любой треугольник можно преобразовать к единичному симплексу.

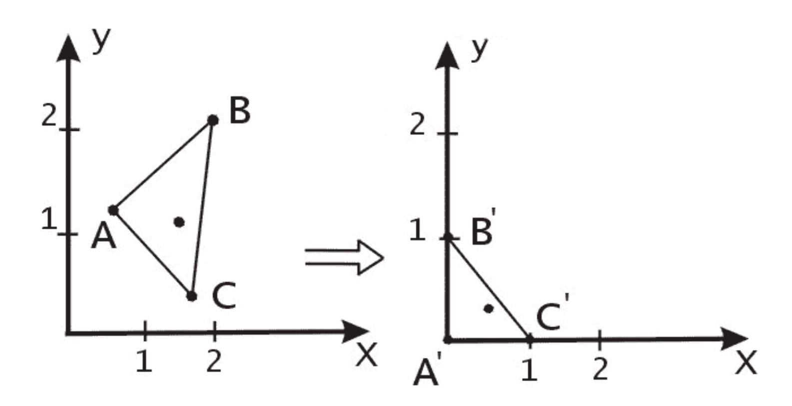


Рис. 13. Приведение произвольного треугольника к единичному симплексу.

После таких преобразований внутренняя и внешняя области треугольника остаются таковыми. Применив такое преобразование к искомой точке, достаточно затем будет определить ее нахождение во внутренней или внешней области симплекса. Найдем такое преобразование. Координаты векторов единичного базиса совпадают с координатами точек Q и R симплекса, соответственно. Будем считать что точка C треугольника совпадает с началом координат. Этого всегда можно добиться параллельным переносом треугольника на вектор $-\overline{C}$. При этом координаты точек A и B треугольника суть коэффициенты разложения соответствующих векторов $\overline{A}$ и $\overline{B}$ по единичному базису. Матрица перехода M от единичного базиса к базису на векторах $\overline{A}$ и $\overline{B}$ составлена из координат этих векторов.

$$M = \begin{bmatrix} A_x & A_y \\ B_x & B_y \end{bmatrix}$$

Значит для обратного перехода к единичному базису, (на векторах которого построен симплекс), необходимо найти обратную матрицу:

$$M^{-1} = \frac{1}{A_x B_y - A_y B_x} \begin{bmatrix} B_y & -A_y \\ -B_x & A_x \end{bmatrix}.$$

Умножение радиус-вектора искомой точки на матрицу M^{-1} дает точку которую достаточно проверить на попадание во внутреннюю или внешнюю область единичного симплекса, как было указано выше.

Проецирование трехмерных объектов

Рассмотрим проблему показа трехмерных изображений на двумерной плоскости. Для этого необходимо иметь определенные математические модели. В этих моделях должны учитываться различные факторы, влияющие на визуальное восприятие человеком реальных образов. Способ перехода от трехмерных объектов к их изображениям на плоскости будем называть проекцией. Далее рассматриваются различные виды проекций.

Для того, чтобы увидеть на плоскости монитора трехмерное изображение нужно уметь задать способ отображения трехмерных точек в двумерные. Сделать это можно, вообще говоря, по-разному. В общем случае проекции преобразуют точки, заданные в системе координат размерностью n в точки системы координат размерностью меньшей, чем n. В нашем случае точки трехмерного пространства преобразуются в точки двумерного пространства. Проекции строятся с помощью проецирующих лучей или проекторов, которые выходят из точки, которая называется центром проекции. Проекторы проходят через плоскость, которая называется проекционной или картинной плоскостью и затем проходят через каждую точку трехмерного объекта и образуют тем самым проекцию. Тип проецирования на плоскую, а не искривленную поверхность, где в качестве проекторов используются прямые. а не искривленные линии, называется плоской геометрической проекцией. Плоские геометрические проекции делятся на два вида: центральные и параллельные. Если центр проекции находится на конечном расстоянии от проекционной плоскости, то проекция – центральная. Если же центр проекции удален на бесконечность, то проекция – параллельная.

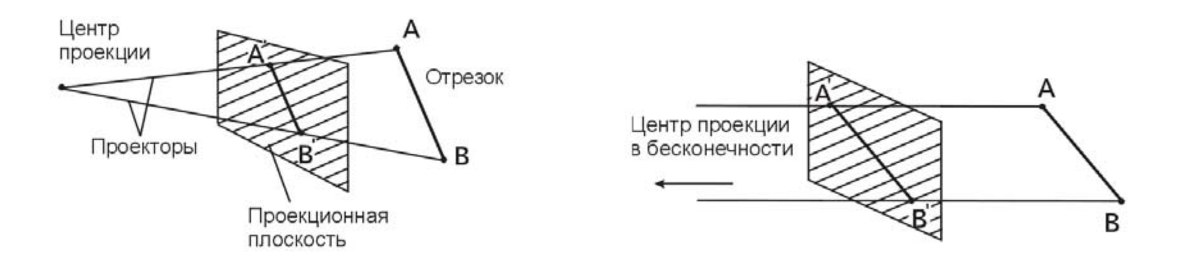


Рис. 14. Центральная проекция. Рис. 15. Параллельная проекция.

Точкой схода называется точка пересечения центральных проекций любой совокупности параллельный прямых, которые не параллельны проекционной плоскости. Существует бесконечное множество точек схода. Точка схода называется главной если совокупность прямых параллельна одной из координатных осей. В зависимости от того, сколько координатных осей пересекает проекционную плоскость различают одно-, двух- и трехточечные проекции.

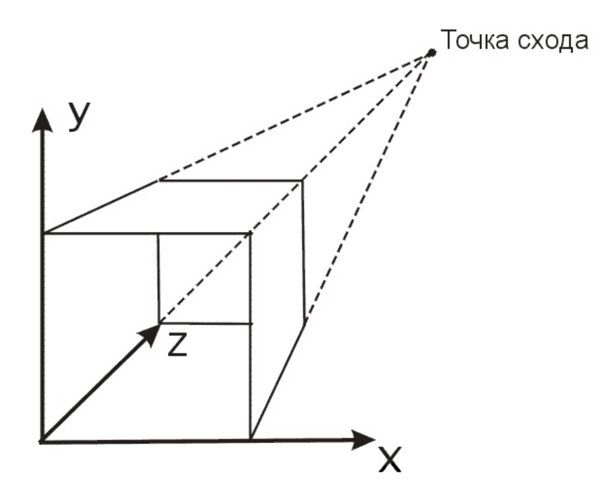


Рис. 16. Одноточечная проекция.

Простейшей является параллельная прямоугольная проекция. В ней совместно изображаются виды сверху, спереди и сбоку. Эти проекции часто используются в черчении. В зависимости от соотношения между направлениями проецирования и нормалью к проекционной плоскости параллельные проекции разделяются на ортографические или ортогональные, в которых эти направления совпадают, и косоугольные, в которых они не совпадают. В зависимости от положения осей системы координат объекта относительно проекционной плоскости ортографические проекции делятся на

аксонометрические и изометрические. В изометрических проекциях оси системы координат составляют одинаковые углы с проекционной плоскостью. В аксонометрических проекциях эти углы разные. Центральная перспективная проекция приводит к визуальному эффекту, подобному тому, который дает зрительная система человека. При этом наблюдается эффект перспективного укорачивания, когда размер проекции объекта изменяется обратно пропорционально расстоянию от центра проекции до объекта. В параллельных проекциях отсутствует перспективное укорачивание, за счет чего изображение получается менее реалистичным и параллельные прямые всегда остаются параллельными.

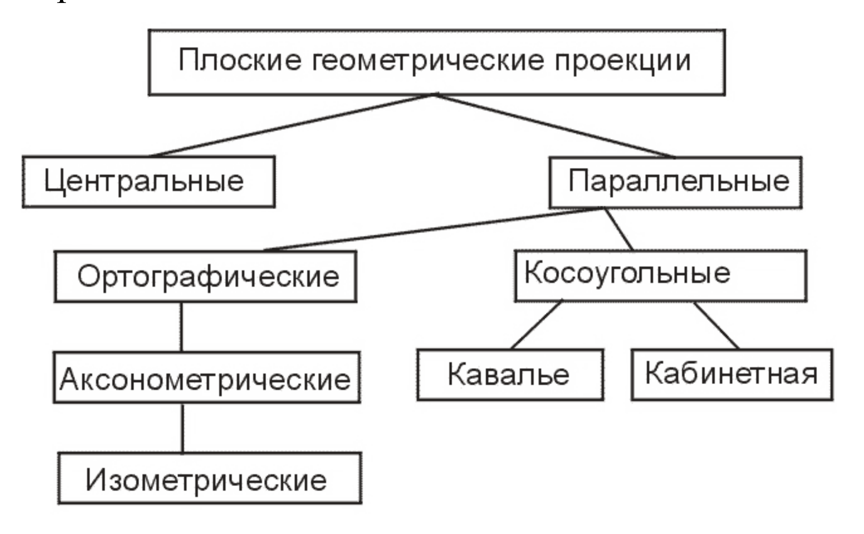


Рис. 17. Типы проекций.

Рассмотрим более подробно центральную перспективную проекцию с математической точки зрения. Для получения формул центральной перспективной проекции расположим оси системы координат, проекционную плоскость и центр проекции как показано на рис. 18.

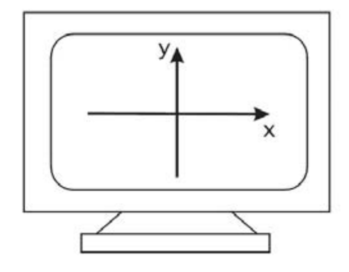


Рис. 18. Расположение осей координат на экране.

Будем имитировать на экране то, что как будто бы реально находится в пространстве за ним. Заметим, что получилась левосторонняя система координат. Будем считать что плоскость экрана монитора совпадает с проекционной плоскостью. Прежде чем переходить к собственно вычислениям следует сделать одно важное замечание. Поскольку поверхность любого трехмерного объекта содержит бесконечное число точек, то необходимо задать способ описания поверхности объекта конечным числом точек для представления в компьютере. А именно, будем использовать линейную аппроксимацию объектов в трехмерном пространстве с помощью отрезков прямых и плоских многоугольников. При этом отрезки прямых после перспективного преобразования переходят в отрезки прямых на проекционной плоскости. Доказательство этого достаточно простое и здесь не приводится. Это важное свойство центральной перспективы позволяет проецировать, т.е. производить вычисления только для конечных точек отрезков, а затем соединять проекции точек линиями уже на проекционной плоскости.

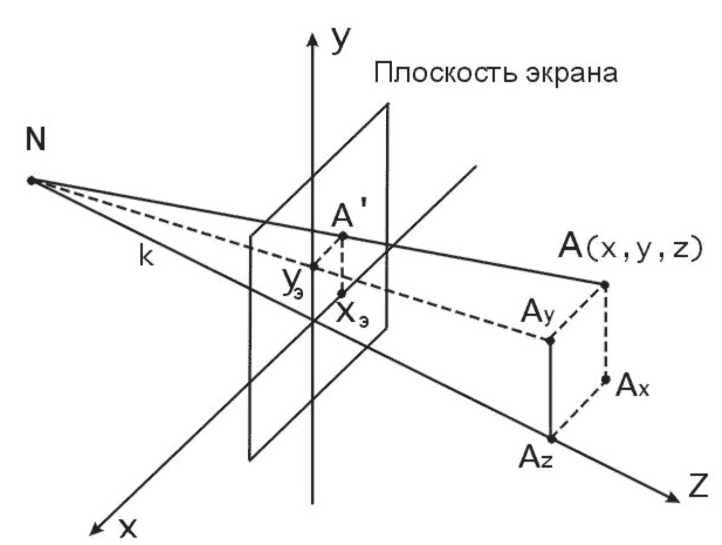


Рис. 19. Вывод формул центральной перспективной проекции.

Точка A проецируется на экран как A'. Расстояние от наблюдателя до проекционной плоскости равно k. Необходимо определить координаты точки A' на экране. Обозначим их $x_э$ и $y_э$. Из подобия треугольников $A_y A_z N$ и $y_э O N$ находим, что

$$\frac{y}{z+k} = \frac{y_э}{k}, \Rightarrow y_э = \frac{ky}{z+k} \qquad (1)$$

аналогично для x: $x_э = \dfrac{kx}{z+k}$.

Напомним, что k -это расстояние, а наблюдатель находится в точке $N = (0,0,-k)$.

Если точку наблюдения поместить в начало координат, а проекционную плоскость на расстояние a, как показано на рисунке 20, то формулы для $x_э$ и $y_э$ примут вид:

$$x_э = \frac{kx}{z}, \qquad y_э = \frac{ky}{z} \tag{2}$$

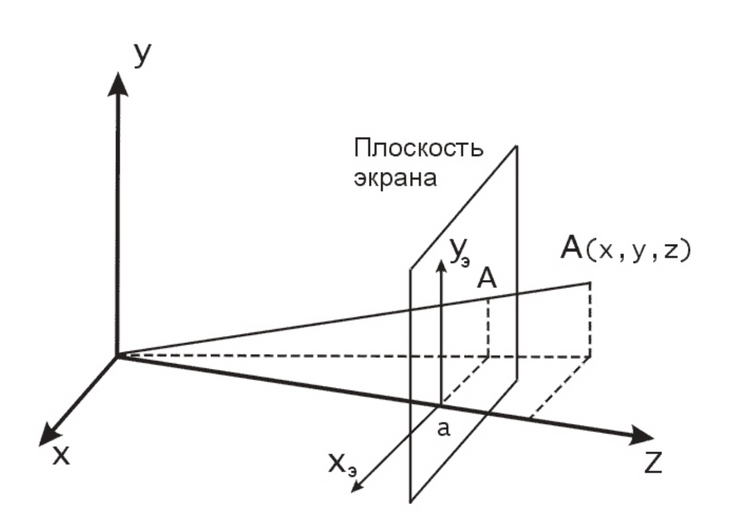


Рис. 20. Другой способ вычисления координат точек в центральной перспективной проекции.

Формулы (1) более удобны при необходимости простым образом приближать или удалять наблюдателя от проекционной плоскости. Формулы (2) требуют меньше времени для вычислений за счет отсутствия операции сложения.

Рассмотрим далее некоторые факторы. влияющие на восприятие человеком трехмерности. Одним из простых способов представления трехмерных объектов являются так называемые проволочные изображения. Кривые линии при этом апроксимируются отрезками прямых. Это наиболее быстрый и простой способ изображения.

Для усиления эффекта трехмерной глубины в проволочных изображениях объектов удаляют невидимые линии. Линии или их части, закрытые поверхностями объекта, не изображаются. Для этого применяется

специальный алгоритм, что требует уже больших вычислений. Передача глубины может осуществляться изменением уровня яркости. Объекты, которые находятся ближе к наблюдателю, изображаются ярче, чем те, которые расположены дальше от него. Движение объектов также дает дополнительный эффект глубины. Например, вращение объектов вокруг вертикальной оси позволяет отличить точки, находящиеся на разном расстоянии от оси за счет различия линейной скорости вращения точек. Это так называемый кинетический или динамический эффект глубины.

Более тонко трехмерность объектов может быть представлена за счет различий отражательных способностей поверхностей, их рельефа и текстуры, а также расчета теней, отбрасываемых поверхностями объекта. Одним из редко используемых, но наиболее эффективных способов достижения эффекта трехмерности является стереоскопия. При этом отдельно для правого и левого глаза наблюдателя формируются изображения, которые незначительно отличаются друг от друга, подобно тому, как это происходит в реальности. Это вызывает так называемый бинокулярный эффект, который заключается в том, что наш мозг сливает два отдельных образа в один, интерпретируемый как трехмерный. Эти два раздельных изображения называются стереопарой.

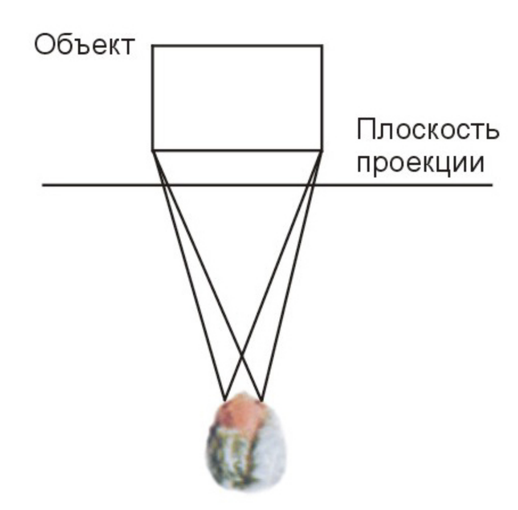


Рис. 21. Бинокулярный эффект, стереоскопия.

Технически этот метод реализуется, например, с помощью очков со специальными поляризованными стеклами. На экран монитора поочередно выводятся изображения для левого и правого глаза. А стекла очков становятся поочередно, соответственно, прозрачными или непрозрачными. При достаточно частой смене изображений смены состояний прозрачности

и непрозрачности не ощущается. Поскольку при изменении положения головы центр проекции остается на месте, то создается псевдо-трехмерный эффект. Синхронизация смены кадров на экране и поляризации линз очков происходит с помощью специальных датчиков, расположенных на очках и мониторе.

Преобразования, связанные с системой координат

Необходимо научиться управлять изображением на экране, вносить изменения в его положение, форму, ориентацию, размер. Для этих целей существуют специальные геометрические преобразования, которые позволяют изменять эти характеристики объектов в пространстве. Представим задачу создания компьютерного имитатора полетов на военном самолете. Объекты на земле, как и сам самолет, изменяют свое положение: вращается антенна локатора, движется танк. При этом, наблюдатель видит эту картину из определенной точки в пространстве в выбранном направлении. Необходимо описать эти сложные преобразования математически.

Введем три вида систем координат. Первая из них – мировая система координат – задается осями $X_M Y_M Z_M$. Мы размещаем ее в некоторой точке, и она остается неподвижной всегда. Вторая – система координат наблюдателя. Эту систему назовем $X_N Y_N Z_N$. Она определяет положение наблюдателя в пространстве и задает направление взгляда. И третья – система координат объекта. В нашем случае их две: система координат локатора и система координат танка. Эти системы также могут перемещаться и изменять свое положение в пространстве относительно мировой системы координат. Координаты точек объектов задаются в системах координат объектов, каждая из которых, в свою очередь, привязана к мировой системе координат. Система координат наблюдателя также перемещается относительно мировой системы координат. Теперь становится понятно, что для того, чтобы увидеть трехмерный объект на экране компьютера надо проделать следующие шаги.

1. Преобразовать координаты объекта, заданные в собственной системе координат, в мировые координаты.
2. Преобразовать координаты объекта, заданные уже в мировой системе координат, в систему координат наблюдателя.
3. Спроецировать полученные координаты на проекционную плоскость в системе координат наблюдателя.

Отметим, определенную двойственность впечатлений, возникающих при взаимных перемещениях систем координат друг относительно друга. Представим себе, что мы наблюдаем кубик в пространстве. Пусть теперь этот кубик начнет вращаться вокруг, например, вертикальной оси. Мы увидим, что кубик вращается. Но тот же самый эффект мы получим, если сами начнем облетать вокруг кубика и рассматривать его с разных сторон. Визуальный эффект остается тем же самым, хотя в первом случае наша система координат остается неподвижной, а во втором – вращается по орбите. Этот эффект можно использовать при выводе формул движения в пространстве.

Двумерные матричные преобразования

Рассмотрим преобразования координат точек на плоскости. На рис. 22 точка A перенесена в точку B.

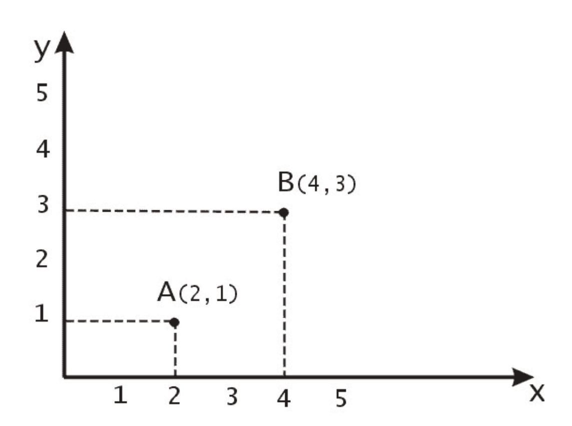


Рис. 22. Операция переноса или трансляции точки A в точку B.

Математически этот перенос можно описать с помощью вектора переноса $\overline{AB}$. Пусть $\overline{R}$ радиус вектор, соответствующий вектору переноса $\overline{AB}$. Тогда переход из точки A в точку B будет соответствовать векторной записи $\overline{B} = \overline{A} + \overline{R}$. Отсюда получаем, что для переноса точки в новое положение необходимо добавить к ее координатам некоторые числа, которые представляют собой координаты вектора переноса:

$$\overline{B} = \overline{A} + \overline{R} = \left[A_x + R_x, A_y + R_y, A_z + R_z\right]$$

Масштабированием объектов называется растяжение объектов вдоль соответствующих осей координат относительно начала координат. Эта

операция применяется к каждой точке объекта, поэтому можно также говорить о масштабировании точки. При этом, конечно, речь не идет об изменении размеров самой точки. Масштабирование достигается умножением координат точек на некоторые константы. В том случае, когда эти константы равны между собой, масштабирование называется однородным. На рис.23 приведен пример однородного масштабирования треугольника ABC.

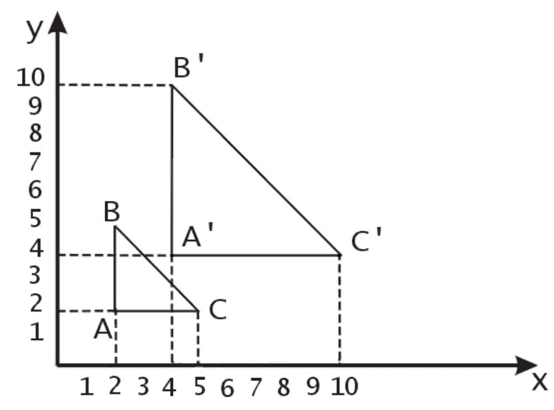


Рис. 23. Операция масштабирования .

После применения операции однородного масштабирования с коэффициентом 2 он переходит в треугольник $A^{'}B^{'}C^{'}$. Обозначим матрицу масштабирования $S = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix}$. Для точек A и $A^{'}$ операция масштабирования в матричном виде будет выглядеть следующим образом:

$$[x', y'] = [x, y]\begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix}.$$

Рассмотрим далее операцию вращения точки на некоторый угол относительно начала координат. На рисунке 24 точка $A = (x, y)$ переходит в точку $B = (x^{'}, y^{'})$ поворотом на угол α.

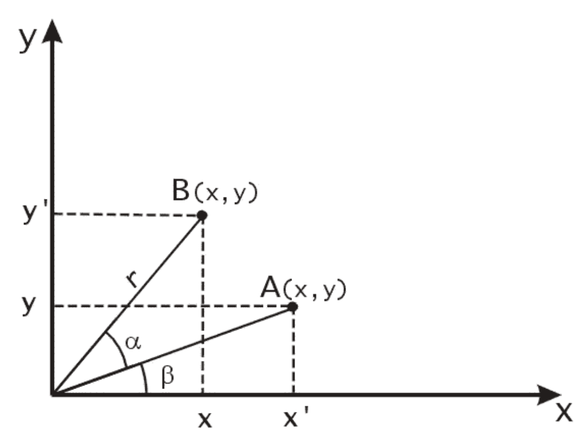


Рис. 24. Операция поворота точки A на угол α.

Найдем преобразование координат точки А в точку В. Обозначим β угол, который составляет радиус-вектор $\overline{A}$ с осью Оx. Пусть r – длина радиус-вектора $\overline{A}$, тогда

$$x' = r \cdot Cos(\alpha + \beta) = r(Cos\alpha \cdot Cos\beta - Sin\alpha \cdot Sin\beta)$$

$$y' = r \cdot Sin(\alpha + \beta) = r(Sin\alpha \cdot Cos\beta + Cos\alpha \cdot Sin\beta)$$

Так как $Cos\beta = x/r$ и $Sin\beta = y/r$, то подставляя эти выражения в уравнения для x' и y', получаем:

$$x' = x \cdot Cos\alpha - y \cdot Sin\alpha$$

$$y' = x \cdot Sin\alpha + y \cdot Cos\alpha$$

В матричном виде вращение точки А на угол α выглядит следующим образом:

$$[x', y'] = [x, y]\begin{bmatrix} Cos\alpha & Sin\alpha \\ -Sin\alpha & Cos\alpha \end{bmatrix}$$

Однородные координаты и матричное представление двумерных преобразований

В предыдущем параграфе были рассмотрены три вида преобразований точек на плоскости. Два из них – операции вращения и масштабирования - описываются в виде произведения матрицы на вектор, а третья – операция

переноса – описывается как сумма двух векторов. В случае последовательного выполнения любой комбинации операций вращения и масштабирования результат легко можно записать в виде произведения матриц соответствующих преобразований. Это будет матрица результирующего поворота и масштабирования. Очевидно, что удобнее применять результирующую матрицу вместо того, чтобы каждый раз заново вычислять произведение матриц. Однако, таким способом нельзя получить результирующую матрицу преобразования, если среди последовательности преобразований присутствует хотя бы один перенос. Матричное произведение в компьютерной графике также называют композицией. Было бы удобнее иметь математический аппарат, позволяющий включать в композиции преобразований все три выше указанные операции. При этом получился бы значительный выигрыш в скорости вычислений. Однородные координаты и есть этот математический аппарат.

Двумерный вектор (x, y) в однородных координатах записывается в виде (wx, wy, w), где $w \neq 0$. Число w называется масштабным множителем. Для того, чтобы из вектора, записанного в однородных координатах получить вектор в обычных координатах необходимо разделить первые две координаты на третью: $(wx / w, wy / w, w / w) \to (x, y, 1)$.

В общем случае осуществляется переход от n-мерного пространства к $(n+1)$-мерному. Это преобразование не единственное. Обратное преобразование называется проекцией однородных координат*.

Рассмотрим некоторые свойства однородных координат. Некоторые точки, неопределенные в n-мерном пространстве, становятся вполне определенными при переходе к однородным координатам. Например, однородный вектор $(0,0,1,0)$ в трехмерном пространстве соответствует бесконечно удаленной точке $z = \infty$. Поскольку в однородных координатах эту точку можно представить в виде $(0,0,1,\varepsilon)$, при $\varepsilon \to 0$, то в трехмерном пространстве это соответствует точке $(0,0,1/\varepsilon)$.

Рассмотрим точку трехмерного пространства (a, b, c). Если представить эту точку как однородное представление точки двумерного пространства, то ее координаты будут $(a/c, b/c)$. Сравнивая эти координаты со вторым видом формул, выведенных для центральной перспективной проекции, легко

---

* Более строгое определение однородных координат дается в разделе линейной алгебры «Проективные пространства».

заметить, что двумерное представление точки с координатами (a,b,c) выглядит как ее проекция на плоскость $z=1$, как показано на рис. 25.

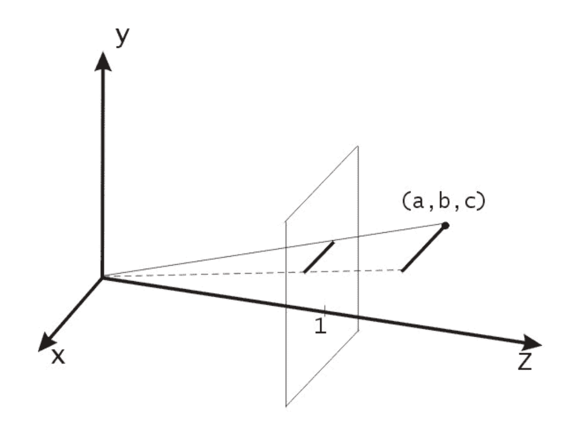


Рис. 25. Проекция точки (a,b,c) на плоскость $z=1$.

Аналогично, рассматривая применение однородных координат для векторов трехмерного пространства, можно представить трехмерное пространство как проекцию четырехмерного пространства на гиперплоскость $w=1$, если $(x,y,z)\to(wx,wy,wz,w)=(x,y,z,1)$.

В однородных координатах преобразование центральной перспективы можно определить матричной операцией. Эта матрица записывается в виде:

$$\begin{bmatrix} k & 0 & 0 & 0 \\ 0 & k & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & k \end{bmatrix} = P$$

Покажем, что эта матрица определяет преобразование точки объекта, заданной в однородных координатах, в точку перспективной проекции (также в однородных координатах). Пусть $p=(x,y,z)$ – точка в трехмерном пространстве. Ее однородное представление $\mathrm{v}=(wx,wy,wz,w)$. Умножим v на P:

$$\mathrm{v}\,P=[wkx,wky,0,w(z+k)]=[kx/(z+k),ky/(z+k),0,1]$$

- это в точности повторяет формулы (1), выведенные для центральной перспективы.

Теперь точки двумерного пространства будут описываться трехэлементными вектор-строками, поэтому и матрицы преобразований, на которые будет умножаться вектор точки, будут иметь размеры 3×3. Запишем матричное преобразование операции переноса для однородных координат:

$$[x', y', 1] = [x, y, 1] \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ D_x & D_y & 1 \end{bmatrix}$$

или $p' = p \cdot T(D_x, D_y)$, где $T(D_x, D_y) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ D_x & D_y & 1 \end{bmatrix}$.

При последовательном переносе точки p в точку p' и затем в точку p'' компоненты суммарного вектора переноса являются суммами соответствующих компонент последовательных векторов переноса. Рассмотрим, каковы будут элементы матрицы суммарного переноса. Пусть $p' = p \cdot T(D_x, D_y)$, $p'' = p' \cdot T(D'_x, D'_y)$. Подставив первое уравнение во второе получаем $p'' = p \cdot T(D_x, D_y) \cdot T(D'_x, D'_y)$. Матричное произведение т.е. суммарный перенос равен произведению соответствующих матриц переноса.

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ D_x & D_y & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ D'_x & D'_y & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ D_x + D'_x & D_y + D'_y & 1 \end{bmatrix}$$

Запишем матричный вид операции масштабирования.

$$[x', y', 1] = [x, y, 1] \cdot \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Определим матрицу масштабирования $S(S_x, S_y) = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Так же, как последовательные переносы являются аддитивными, покажем, что последовательные масштабирования будут мультипликативными.

$$S(S_x, S_y)\cdot S(S'_x, S'_y) = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} S'_x & 0 & 0 \\ 0 & S'_y & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} S_x \cdot S'_x & 0 & 0 \\ 0 & S_y \cdot S'_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Для операции поворота матричный вид будет такой:

$$[x', y', 1] = [x, y, 1]\cdot \begin{bmatrix} Cos\alpha & Sin\alpha & 0 \\ -Sin\alpha & Cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Определим матрицу поворота $R(\alpha) = \begin{bmatrix} Cos\alpha & Sin\alpha & 0 \\ -Sin\alpha & Cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Аналогично двум предыдущим случаям, покажем, что матрица поворота остается таковой при последовательных поворотах.

$$R(\alpha)R(\beta) = \begin{bmatrix} Cos\alpha & Sin\alpha & 0 \\ -Sin\alpha & Cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} Cos\beta & Sin\beta & 0 \\ -Sin\beta & Cos\beta & 0 \\ 0 & 0 & 1 \end{bmatrix} =$$

$$= \begin{bmatrix} Cos(\alpha+\beta) & Sin(\alpha+\beta) & 0 \\ -Sin(\alpha+\beta) & Cos(\alpha+\beta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Таким образом, доказано, что два, а значит и любое количество последовательных поворотов можно записать в виде одной матрицы суммарного поворота. Также легко заметить что любая последовательность операций, включающая в себя перенос, масштабирование и вращение в однородных координатах, может быть представлена одной матрицей, которая является произведением матриц данных операций.

Рассмотрим, каким образом с помощью композиции матричных преобразований можно получить одно общее результирующее преобразование. Для этого будем использовать матрицы T, S и R. С вычислительной точки зрения гораздо проще и быстрее применять матрицу уже готового преобразования вместо того, чтобы применять их

последовательно одну за другой. К точке более эффективно применять одно результирующее преобразование, чем ряд преобразований друг за другом.

Для примера рассмотрим задачу поворота объекта на плоскости относительно некоторой произвольной точки p_0. Пока мы умеем поворачивать объекты только вокруг начала координат. Но можно представить эту задачу как последовательность шагов, на каждом из которых будет применяться только элементарная операция: перенос, масштабирование или вращение.

Вот эта последовательность элементарных преобразований (рис. 26):

1. Перенос, при котором точка p_0 переходит в начало координат.
2. Поворот на заданный угол.
3. Перенос, при котором точка из начала координат возвращается в первоначальное положение p_0.

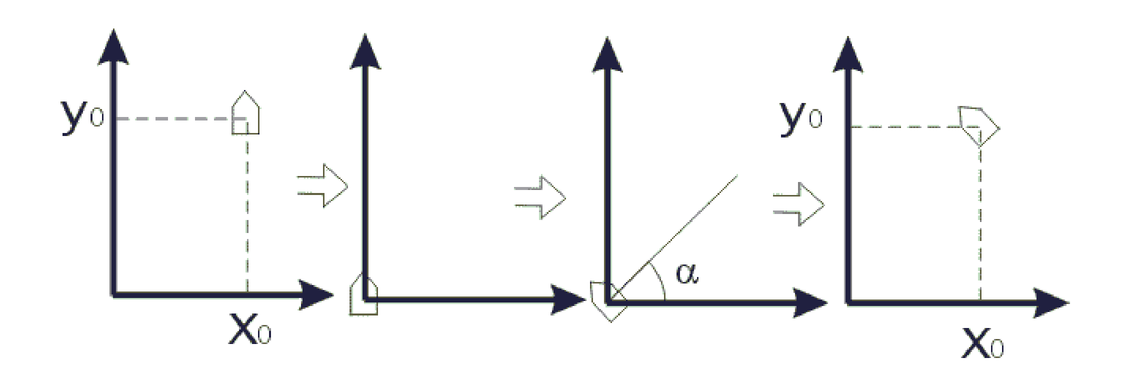


Рис. 26. Последовательность преобразований при повороте объекта вокруг точки $p_0 = (x_0, y_0)$ на угол α.

Точка $p_0 = (x_0, y_0)$. Первый перенос производится на вектор $[-x_0, -y_0]$, а обратный перенос - на вектор $[x_0, y_0]$.

Трехмерные матричные преобразования

Подобно тому, как двумерные преобразования описываются матрицами размером 3×3, трехмерные преобразования могут быть представлены

матрицами размером 4×4. Тогда трехмерная точка (x, y, z) записывается в однородных координатах как (wx, wy, wz, w), где $w \neq 0$. Для получения декартовых координат надо первые три однородные координаты разделить на w. Два однородных вектора описывают одну декартову точку в трехмерном пространстве, если $H_1 = cH_2$, где $c = Const \neq 0$ и H_1, H_2 - векторы, записанные в однородных координатах.

Матрицы преобразований будем записывать в правосторонней системе координат. При этом положительный поворот определяется следующим образом. Если смотреть из положительной части оси вращения (например, оси z) в направлении начала координат, то поворот на 90^o против часовой стрелки будет переводить одну положительную полуось в другую (ось x в y, в соответствии с правилом циклической перестановки).

Заметим, что на практике удобнее применять левостороннюю систему координат, так как в этом случае удобнее интерпретировать тот факт, что точки с большими значениями z находятся дальше от наблюдателя.

Запишем теперь матрицу трехмерного переноса. Аналогично двумерному случаю.

$$T(D_x, D_y, D_z) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ D_x & D_y & D_z & 1 \end{bmatrix}, \text{ при этом}$$

$$[x, y, z, 1] \cdot T(D_x, D_y, D_z) = [x + D_x, y + D_y, z + D_z, 1].$$

Операция масштабирования:

$$S(S_x, S_y, S_z) = \begin{bmatrix} S & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$[x, y, z, 1] \cdot S(S_x, S_y, S_z) = [S_x \cdot x, S_y \cdot y, S_z \cdot z, 1]$$

Перейдем к операции поворота, с ней в трехмерном случае придется разбираться чуть побольше чем в двумерном. Так как при двумерном повороте в плоскости xy координаты z остаются неизменными, то поворот вокруг оси z записывается так:

$$R_z(\alpha)=\begin{bmatrix} Cos\alpha & Sin\alpha & 0 & 0 \\ -Sin\alpha & Cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Матрица поворота вокруг оси x имеет вид:

$$R_x(\alpha)=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & Cos\alpha & Sin\alpha & 0 \\ 0 & -Sin\alpha & Cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

и вокруг оси y:

$$R_y(\alpha)=\begin{bmatrix} Cos\alpha & 0 & -Sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ Sin\alpha & 0 & Cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Обратите внимание на смену положения синуса угла с отрицательным знаком в матрице поворота вокруг оси y. Правильность этих матриц легко проверить поворотом одного из ортов на 90^{o}, при этом он должен перейти в следующий по порядку орт на соответствующей координатной оси.

Обратные преобразования будут выражаться обратными матрицами. Для операции переноса надо лишь заменить знаки компонент вектора переноса на противоположные:

$$T^{-1}(D_z, D_y, D_z)=T(-D_z, -D_y, -D_z);$$

для операции масштабирования – на обратные значения:

$$S^{-1}(S_z, S_y, S_z)=S(1/S_z, 1/S_y, 1/S_z);$$

для поворота – выбором отрицательного угла поворота:

$$R^{-1}(\alpha)=R(-\alpha).$$

Результатом нескольких последовательных поворотов будет матрица

$$A=\begin{bmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Здесь верхняя матрица размером 3×3 называется ортогональной. Важным ее свойством является то, что обратная к ней матрица является транспонированной: $B^{-1} = B^T$. Это полезно тем, что при вычислениях достаточно поменять индексы местами и обратное преобразование получается автоматически.

После перемножения любого числа матриц вида T, S и R результирующая матрица всегда будет иметь вид:

$$\begin{bmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ t_x & t_y & t_z & 1 \end{bmatrix}.$$

Здесь верхняя часть размером 3×3 определяет суммарный поворот и масштабирование, а три коэффициента последней строки – суммарный перенос.

Вопросы эффективности вычислений

Рассмотрим проблему ускорения вычислений в одной из самых трудоемких операций компьютерной графики – операции поворота точки относительно начала координат. Как было показано ранее, для ее выполнения необходимо произвести 4 операции умножения, 2 операции сложения, а также вычислить значения синуса и косинуса угла поворота. Напомним вид формул поворота:

$$x' = x \cdot Cos\alpha - y \cdot Sin\alpha$$

$$y' = x \cdot Sin\alpha + y \cdot Cos\alpha$$

Одним из наиболее часто встречающихся способов ускорения операции поворота является отказ от вычисления синуса и косинуса угла во время выполнения программы, и использование их заранее подсчитанных значений, которые занесены в специальную таблицу. Например, в этой таблице могут храниться значения синусов и косинусов углов поворота с шагом в 1 градус. Тогда целое количество градусов угла поворота может служить в качестве индекса при извлечении соответствующих значений синусов и косинусов из таблицы. Такой прием называется табличным поворотом.

Дополнительным способом ускорения операции поворота является уменьшение количества операций умножения. Рассмотрим вывод формулы О. Бьюнемана с использованием тангенса половинного угла, в которой

поворот точки вокруг начала координат производится за 3 операции умножения и 3 операции сложения. Так как на многих микропроцессорах операции умножения выполняются дольше чем операции сложения, то экономия времени достигается за счет уменьшения операций умножения.

Вывод формулы будем получать из геометрических построений, как показано на рис.27.

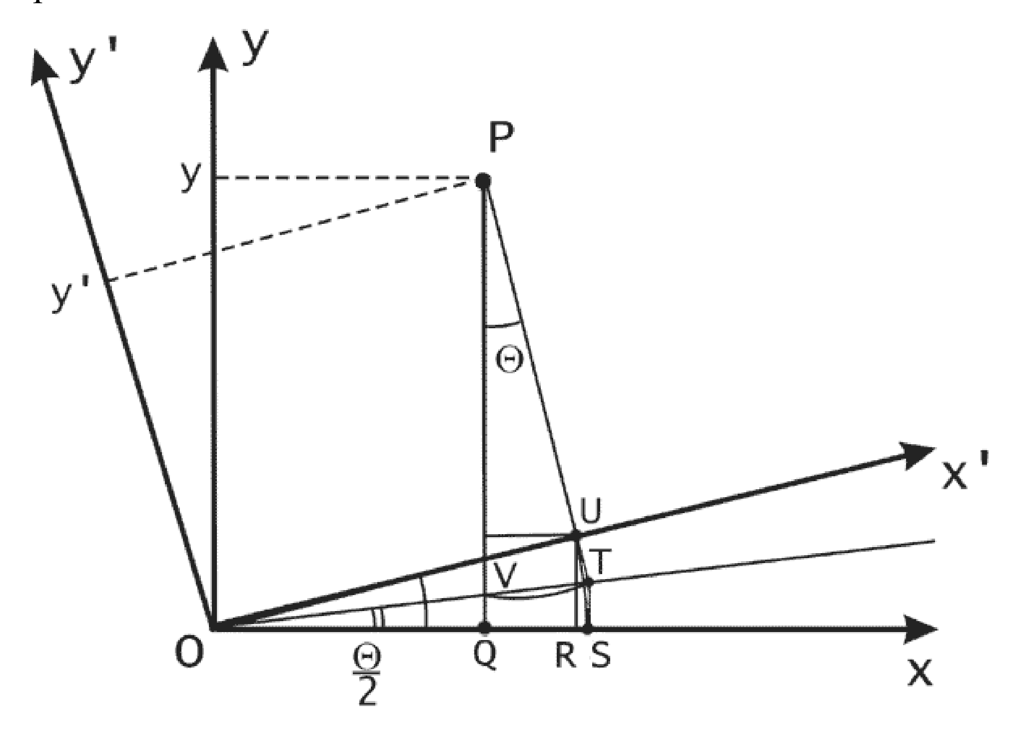


Рис. 27. Вывод формулы О. Бьюнемана.

Будем искать выражение координат x и y через x' и y'. На оси Ox отложим отрезок OS, такой что $OS = x'$. Тогда $x = OS - QS = x' - QS$. Здесь отрезок QS является горизонтальной проекцией отрезка PT, где $PT = PU + UT$, $PU = y'$, $UT = x' tg\frac{\theta}{2}$ $\Rightarrow x = x' - PT \cdot Sin\theta$, где $PT = y' + x' tg\frac{\theta}{2}$. Теперь, зная x, можно выразить y в виде суммы длин отрезков QV и VP. Так как длины отрезков PV и PT равны как радиусы окружности с центром в точке P, то $y = x \cdot tg\frac{\theta}{2} + PT$. Обозначим $PT = T^*$, отсюда следует, что

$$T^* = y' + x' tg\frac{\theta}{2},$$
$$x = x' - T^* Sin\theta,$$
$$y = x \cdot tg\frac{\theta}{2} + T^*$$

Последние три равенства будем называть формулой Бьюнемана.

Алгоритмы растровой графики

Растром называется прямоугольная сетка точек, формирующих изображение на экране компьютера. Каждая точка растра характеризуется двумя параметрами: своим положением на экране и своим цветом, если монитор цветной, или степенью яркости, если монитор черно-белый. Поскольку растровые изображения состоят из множества дискретных точек, то для работы с ними необходимы специальные алгоритмы. Рисование отрезка прямой линии - одна из простейших задач растровой графики. Смысл ее заключается в вычислении координат пикселов, находящихся вблизи непрерывных отрезков, лежащих на двумерной растровой сетке.

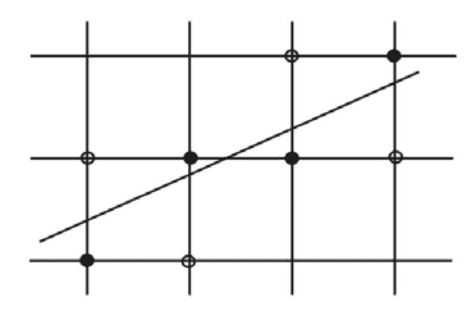


Рис. 28. Растеризация отрезка прямой линии.

Термин “пиксел” образован от английского pixel (picture element - элемент изображения) - то есть точка на экране. Будем считать, что пикселы имеют целочисленные координаты. На первый взгляд кажется, что эта задача имеет простое решение. Пусть конечные точки отрезка имеют целочисленные координаты, и уравнение прямой, содержащей отрезок: $y = kx + b$. Не нарушая общности, будем также считать, что тангенс угла наклона прямой лежит в пределах от 0 до 1. Тогда для изображения отрезка на растре

достаточно для всех целых x, принадлежащих отрезку, выводить на экран точки с координатами $(x, \boldsymbol{Round}(y))$. Однако в этом методе присутствует операция умножения kx. Хотелось бы иметь алгоритм без частого использования операции умножения вещественных чисел. Избавиться от операции умножения можно следующим образом. Поскольку $k = \Delta y / \Delta x$, то один шаг по целочисленной сетке на оси x будет соответствовать $\Delta x = 1$. Отсюда получаем, что y будет увеличиваться на величину k. Итерационная последовательность выглядит следующим образом:

$$x_{i+1} = x_i + 1, \quad y_{i+1} = y_i + k$$

Когда $k > 1$, то шаг по x будет приводить к шагу по $y > 1$, поэтому x и y следует поменять ролями, придавая y единичное приращение, а x будет увеличиваться на $\Delta x = \Delta y / k = 1/k$ единиц. Этот алгоритм все же не свободен от операций с вещественными числами. Наиболее изящное решение задачи растровой развертки отрезков прямых было найдено Брезенхемом. В его алгоритме вообще не используются операции с вещественными числами, в том числе операции умножения и деления.

Для вывода формул алгоритма Брезенхема рассмотрим рис. 29.

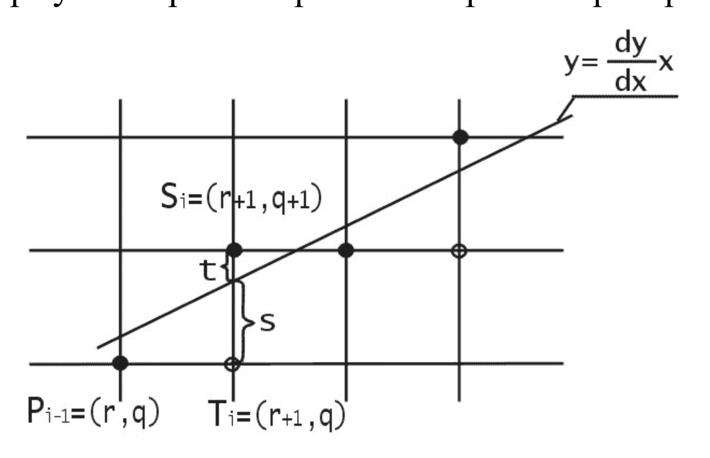


Рис. 29. Рисование отрезков прямых по методу Брезенхема.

Пусть начало отрезка имеет координаты $(x1, y1)$, а конец $(x2, y2)$. Обозначим $dx = (x2 - x1)$, $dy = (y2 - y1)$. Не нарушая общности, будем считать, что начало отрезка совпадает с началом координат, и прямая имеет вид $y = \frac{dx}{dy} x$, где $\frac{dx}{dy} \in [0,1]$. Считаем что начальная точка находится слева.

Пусть на $(i-1)$-м шаге текущей точкой отрезка является $P_{i-1}=(r,q)$. Выбор следующей точки S_i или T_i зависит от знака разности $(s-t)$. Если $(s-t)<0$, то $P_i=T_i=(r+1,q)$ и тогда $x_{i+1}=x_i+1$, $y_{i+1}=y_i$, если же $(s-t)\ge 0$, то $P_i=S_i=(r+1,q+1)$ и тогда $x_{i+1}=x_i+1$, $y_{i+1}=y_i+1$.

$$s=\frac{dy}{dx}(r+1)-q\,,\ t=q+1-\frac{dy}{dx}(r+1),\Rightarrow$$

$$s-t=2\frac{dy}{dx}(r+1)-2q-1\ \Rightarrow$$

$$dx(s-t)=2(r\cdot dy-q\cdot dx)+2dy-dx\,.$$

Поскольку знак $dx(s-t)$ совпадает со знаком разности $(s-t)$, то будем проверять знак выражения $d_i=dx(s-t)$. Так как $r=x_{i-1}$ и $q=y_{i-1}$, то $d_{i+1}=d_i+2dy-2dx(y_i-y_{i-1})$.

Пусть на предыдущем шаге $d_i<0$, тогда $(y_i-y_{i-1})=0$ и $d_{i+1}=d_i+2dy$. Если же на предыдущем шаге $d_i\ge 0$, то $(y_i-y_{i-1})=1$ и $d_{i+1}=d_i+2(dy-dx)$.

Осталось узнать как вычислить d_1. Так как при $i=1$:

$$(x_0,y_0)=(0,0),\ \Rightarrow d_1=2dy-dx\,.$$

Далее приводится листинг процедуры на языке Паскаль, реализующей алгоритм Брезенхема.

```
Procedure Bresenham(x1,y1,x2,y2,Color: integer);
var
dx,dy,incr1,incr2,d,x,y,xend: integer;
begin
  dx:= ABS(x2-x1);
  dy:= Abs(y2-y1);
  d:=2*dy-dx;        {начальное значение для d}
  incr1:=2*dy;       {приращение для d<0}
  incr2:=2*(dy-dx);  {приращение для d>=0}
   if x1>x2 then     {начинаем с точки с меньшим знач. x}
      begin
       x:=x2;
       y:=y2;
       xend:=x1;
     end
```

```
     else
     begin
       x:=x1;
       y:=y1;
       xend:=x2;
     end;
 PutPixel(x,y,Color);    {первая точка отрезка}
 While x<xend do
  begin
     x:=x+1;
   if d<0 then
     d:=d+incr1          {выбираем нижнюю точку}
   else
     begin
      y:=y+1;
      d:=d+incr2;   {выбираем верхнюю точку, y-возрастает}
     end;
   PutPixel(x,y,Color);
  end;{while}
end;{procedure}
```

Перед тем, как исследовать методы получения изображений более сложных, чем отрезки прямых, рассмотрим проблему, незримо присутствующую в большинстве задач компьютерной графики. Эта проблема отсечения изображения по некоторой границе, например, по границе экрана, или, в общем случае, некоторого прямоугольного окна. Рассмотрим эту задачу применительно к отрезкам прямых. Некоторые из них полностью лежат внутри области экрана, другие целиком вне ее, а некоторые пересекают границу экрана. Правильное отображение отрезков означает нахождение точек пересечения их с границей экрана и рисование только тех их частей, которые попадают на экран. Один из очевидных способов отсечения отрезков состоит в определении точек пересечения прямой, содержащей отрезок, с каждой из четырех прямых, на которых лежат границы окна и проверки не лежит ли хотя бы одна точка пересечения на границе. В этом случае для каждой пары сторона-отрезок необходимо решать систему из двух уравнений, используя операции умножения и деления. При этом удобно параметрическое задание прямых:

$$x = x1 + t(x2 - x1)$$
$$y = y1 + t(y2 - y1).$$

Для $t \in [0,1]$ эти уравнения определяют точки, находящиеся между $(x1, y1)$ и $(x2, y2)$. Специальной проверки требует случай, когда отрезок параллелен стороне окна. Пусть координата x точки пересечения найдена, тогда

$$t = \frac{x - x_1}{x_2 - x_1} \Rightarrow y = y_1 + \frac{x - x_1}{x_2 - x_1}(y_2 - y_1)$$

Рассмотрим алгоритм Коэна-Сазерленда для отсечения отрезков прямых. Этот алгоритм позволяет легко определять нахождение отрезка полностью внутри или полностью снаружи окна, и если так, то его можно рисовать или не рисовать, не заботясь об отсечении по границе окна.

Для работы алгоритма вся плоскость в которой лежит окно разбивается на девять подобластей или квадрантов, как показано на рис. 30.

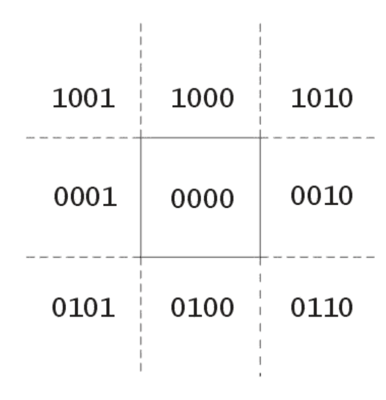


Рис. 30. Разбиение на подобласти в методе Коэна-Сазерленда.

Окну соответствует область обозначенная кодом 0000. Конечным точкам отрезка приписывается 4-битный код “вне/внутри” в зависимости от нахождения отрезка в соответствующей подобласти. Каждому биту присваивается значение 1 в соответствии со следующим правилом.

Бит 1 - точка находится выше окна;

Бит 2 – точка находится ниже окна;

Бит 3 - точка находится справа от окна;

Бит 4 - точка находится слева от окна;

Иначе биту присваивается нулевое значение. Значения этих битов для конечных точек отрезков легко определить по знакам соответствующих разностей: $(y_{max} - y)$ - для 1-го бита, $(y - y_{min})$ - для 2-го бита, $(x_{max} - x)$ - для 3-го бита и $(x - x_{min})$ - для 4-го бита. Отрезок рисуется без отсечения, то есть принимается целиком, если оба кода равны 0000, или $[кодP1]$ИЛИ$[кодP2] = 0000$, где ИЛИ – бинарная операция. Отрезок отбрасывается без вычислений если оба его конца находятся выше, ниже, правее или левее окна. В этих случаях соответствующие биты в обоих кодах равны 1 и это легко определить, умножив эти коды по бинарной операции И. Если результат операции И равен 0000, то отрезок нельзя ни принять ни отбросить, так как он может пересекаться с окном. В этом случае применяется последовательное разделение отрезка, так что на каждом шаге конечная точка отрезка с ненулевым кодом вне/внутри заменяется на точку, лежащую на стороне окна или на прямой содержащей сторону. При этом порядок перебора сторон окна не имеет значения.

Далее приводится текст процедуры на языке Паскаль, с довольно изящной реализацией этого метода. Отрезок задан граничными точками $P1 = (x1, y1)$, $P2 = (x2, y2)$, границы окна: xmin, xmax, ymin, ymax. Используются вызовы процедур: Accept_Check – выполняет проверку на полное принятие отрезка; Reject_Check – на полный отказ от рисования отрезка; Outcodes – вычисляет 4-х битовый код “вне/внутри”; SWAP – меняет местами координаты двух точек.

```
Procedure CLIP(x1,x2,y1,y2,xmin,xmax,ymin,ymax: real);
type
 outcode = array[1..4] of boolean;
var
 accept,reject,done: boolean;
outcode1,outcode2,
outcode3,outcode4:outcode;{коды вне/внутри}
begin
 accept:= false;
 reject:= false;
 done:= false;
repeat
Outcodes(x1,y1,outcode1);
Outcodes(x2,y2,outcode2);
{проверка на отбрасывание}
```

```
 reject:=Reject_Check(outcode1,outcode2);
  if reject then done:= true
  else
     begin {возможно принятие целиком}
          accept:=Accept_Check(outcode1,outcode2);
     if accept then done:=true
     else
begin {разделить отрезок}
{если P1 внутри, то с помощью SWAP сделать снаружи}
if not((outcode1[1])or(outcode1[2])or
       (outcode1[3])or(outcode1[4])) then SWAP;
{теперь P1 перемещается в точку пересечения}
  if outcode1[1] then
   begin {отбросить верхнюю часть}
     x1:=x1+(x2-x1)*(ymax-y1)/(y2-y1);
     y1:=ymax;
   end
else if outcode1[2] then
if outcode1[1] then
   begin {отбросить нижнюю часть}
     x1:=x1+(x2-x1)*(ymin-y1)/(y2-y1);
     y1:=ymin;
   end
else if outcode1[3] then
   begin {отбросить правую часть}
     y1:=x1+(y2-y1)*(ymax-x1)/(x2-x1);
     x1:=xmax;
   end
else if outcode1[4] then
   begin {отбросить левую часть}
     y1:=x1+(y2-y1)*(ymin-x1)/(x2-x1);
     x1:=xmin;
   end;
  end;
 end;
until done;
  if accept then
   Line(x1,y1,x2,y2); {нарисовать отрезок}
end;{procedure}
```

Нормирующие преобразования видимого объема

Зададим центральную перспективную проекцию с центром проекции в начале координат, как показано на рис. 31. Для реальных вычислений необходимо также определить значения минимальной и максимальной отсекающих плоскостей по координате z: $z = z_{\min}$ и $z = z_{\max}$, соответственно.

Границы экрана, или окна вывода задают четыре отсекающих плоскости сверху, снизу, справа и слева. Таким образом, изображение, получаемое с помощью нашей проекции может находится только внутри усеченной пирамиды образованной упомянутыми плоскостями, причем объекты вне этой пирамиды не проецируются на экран, т.е. являются невидимыми для наблюдателя. Видимым объемом называется замкнутая область пространства, объекты внутри которой проецируются на экран. В случае центральной перспективной проекции видимым объемом является усеченная пирамида.

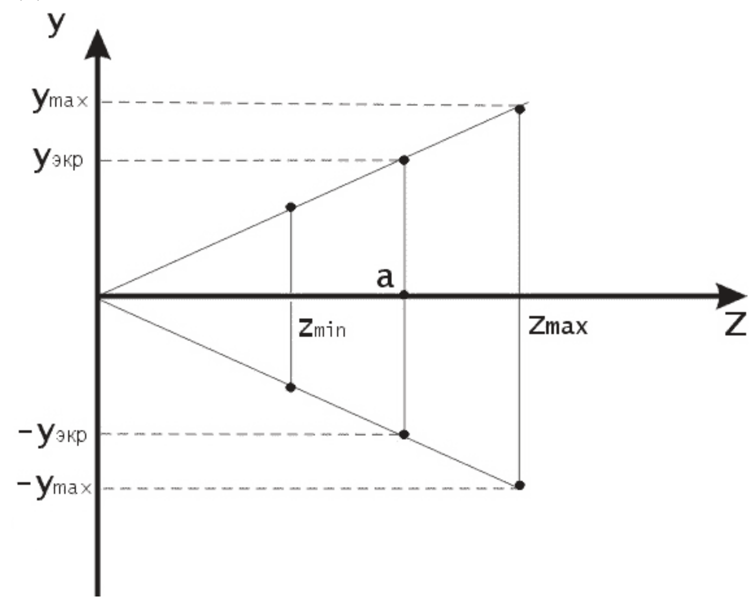


Рис 31. Видимый объем, вид сбоку.

Одной из важных задач компьютерной графики является нахождение эффективного способа отсечения трехмерных объектов по границе видимого объема и удаление невидимых ребер и граней. Например, в случае центральной перспективы, для решения задачи отсечения пришлось бы для каждой грани или ребра находить точки пересечения с плоскостями усеченной пирамиды, что в общем случае потребовало бы значительных вычислений. Решение заключается в преобразовании видимого объема к

виду, в котором вычисления проводились бы значительно проще. В общем идея заключается в том, чтобы свести преобразование центральной перспективы математически к виду параллельной проекции, в которой, очевидно, операция взятия проекции сводится к простому отбрасыванию у точек координаты z.

Будем решать задачу в два этапа. В начале приведем видимый объем к нормированному виду. При этом значение $z_{\max} = 1$, а границы по осям x и y лежат в диапазоне $[-1,1]$, как показано на рис. 32.

Нормирующим преобразованием в этом случае будет операция масштабирования, которая для произвольной точки X выражается в виде:

$$X' = X \cdot S\left(1/x_{\max}, 1/y_{\max}, 1/z_{\max} \right),$$

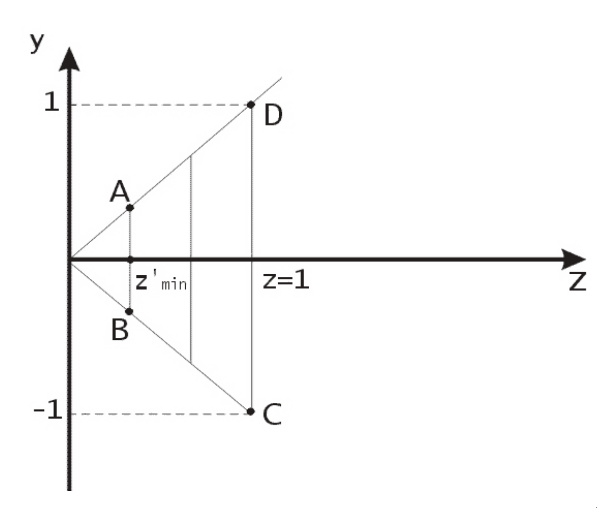


Рис. 32. Нормированный видимый объем.

где $\frac{x_{\max} a}{z_{\max}} = x_{экр} \Rightarrow x_{экр} = \frac{z_{\max} x_{экр}}{a}$, и соответственно, $y_{экр} = \frac{z_{\max} y_{экр}}{a}$.

Нормированный видимый объем позволяет с большей легкостью решать задачу отсечения по границе. А именно, в этом случае может применяться модифицированный вариант алгоритма Коэна-Сазарленда в котором вместо 4-битовых используются 6-битовые коды вне/внутри для описания нахождения точки в соответствующей области пространства. Уравнения боковых граней видимого объема сильно упрощаются, например, для правой отсекающей плоскости уравнение запишется $z = x$, а для левой боковой $z = -x$ и т.д. . Тогда для некоторой точки (x, y, z) условие установления бита в единицу будет следующим:

1-й бит: $y > z$

2-й бит: $y < -z$

3-й бит: $x > z$

4-й бит: $x < -z$

5-й бит: $z > 1$

6-й бит: $z < z'_{\min}$

Для эффективного решения задачи удаления невидимых ребер/граней преобразуем нормированный видимый объем к каноническому виду, как показано на рис. 33.

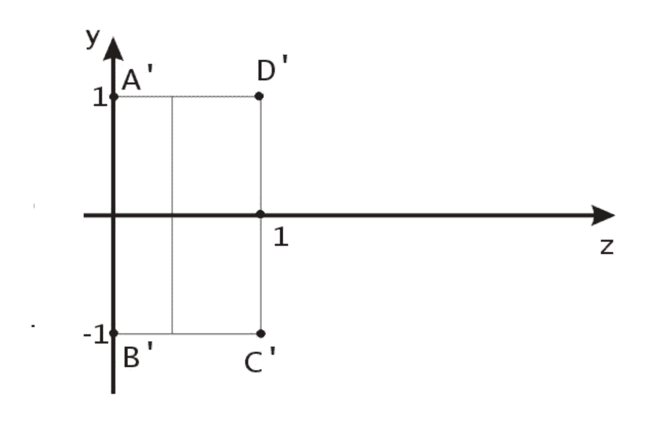


Рис. 33. Канонический видимый объем.

Это достигается с помощью матрицы

$$M_p = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \dfrac{1}{1 - z'_{\min}} & 1 \\ 0 & 0 & \dfrac{-z'_{\min}}{1 - z'_{\min}} & 0 \end{bmatrix}.$$

После применения матрицы M_p нормированный видимый объем становится прямоугольным параллелепипедом, что позволяет перейти от центральной перспективной к параллельной проекции. Легко проверить, что

как показано на рис. 32, 33: $D' = DM_p$, $A' = AM_p$, $B' = BM_p$, $C' = CM_p$, а также, например, $[z'_{min}, z'_{min}, z'_{min}, 1] \xrightarrow{M_p} [1,1,0,1]$.

Итак, нормирующие преобразования видимого объема могут производиться за два шага.

1 шаг - преобразование к нормированному видимому объему и отсечение по 3-х мерному алгоритму Коэна-Сазерленда.

2 шаг - преобразование к прямоугольному параллелепипеду с помощью матрицы M_p и удаление скрытых поверхностей при условии равенства координат x и y.

Алгоритмы удаления невидимых ребер и граней

Алгоритмы удаления невидимых граней могут быть условно поделены на два класса в зависимости от принципов, заложенных для их реализации. Первый класс – это алгоритмы работающие в пространстве объекта. Это означает, что для определения видимости данной грани сравнивается ее взаимное расположение со всеми остальными гранями в трехмерной сцене. Пусть N – количество граней в трехмерной сцене. Для построения трехмерной сцены в этом случае необходимо сравнить положение каждой грани с оставшимися, что требует порядка N^2 операций. Например, пусть количество граней в трехмерной сцене $N = 1000$, тогда время работы алгоритмов этого класса порядка 1,000,000 операций.

Другой класс алгоритмов - работающих в пространстве изображения, основан на нахождении точки ближайшей грани которую пересекает луч зрения, проходящий через заданную точку на растре. Поскольку число точек на растровом экране фиксировано, то алгоритмы этого класса менее чувствительны к увеличению количества объектов в трехмерной сцене. Пусть n - число точек на растровом экране. Тогда количество операций, необходимых для построения трехмерной сцены будет порядка $n \cdot N$. Например, для экранного разрешения 320×200 точек, $n = 64000$, тогда количество операций для $N = 1000$ граней будет порядка 64,000,000. Выбор класса алгоритма может зависеть от особенностей конкретной задачи, а также от способов реализации алгоритма.

Рассмотрим алгоритм удаления невидимых граней с использованием z-буфера, который является одним из наиболее часто используемых в современных приложениях компьютерной графики. Он работает в

пространстве изображения и применяется в таких популярных графических библиотеках как OpenGL и Direct3D.

Алгоритм работает в параллельной проекции. Пусть размеры окна вывода или экрана составляют X точек в ширину и Y точек в высоту. В качестве z-буфера заведем двумерный прямоугольный массив чисел по размерности совпадающий с окном вывода или экрана, т.е. $X \times Y$. В z-буфере будут храниться текущие значения z-координат каждого пиксела.

В начале работы алгоритма в z-буфер заносятся значения, соответствующие бесконечности. Каждая грань трехмерного объекта, представленная в виде многоугольника, преобразуется в растровую форму. При разложении в растр для каждой точки многоугольника вычисляется значение ее z-координаты. Если z-координата оказалась меньше чем текущее значение в z-буфере, то в z-буфер заносится z-координата точки, и на экране рисуется точка цветом текущего многоугольника. После разложения в растр всех многоугольников изображение трехмерной сцены построено.

Рассмотрим способ ускоренного вычисления z-координат при разложении многоугольников в растр. Запишем уравнение плоскости, образуемой многоугольником в пространстве: $Ax + By + Cz + D = 0$.

Выразим z-координату точки: $z = \frac{-D - Ax - By}{C} = f(x, y)$. Пусть $z_0 = f(x_0, y_0)$. Найдем z-координату для соседней точки $z_1 = f(x_0 + \Delta x, y_0) = \frac{-D - A(x_0 + \Delta x) - By_0}{C} = \frac{-D - Ax_0 - By_0}{C} - \frac{A\Delta x}{C} =$ $= z_0 - \frac{A\Delta x}{C}$. Для соседнего пиксела на экране $\Delta x = 1$, тогда $\frac{A\Delta x}{C} = Const$, отсюда следует, что $z_1 = z_0 - Const$. Таким образом, вычисление z-координаты соседнего пиксела сводится к одной операции вычитания.

Рассмотрим далее алгоритм удаления невидимых граней методом сортировки по глубине (авторы: Ньюэлл, Ньюэлл, Санча). Часть этого метода работает в пространстве объекта, а часть в пространстве изображения. Он также работает для параллельной проекции, то есть с учетом того что произведено перспективное преобразование. Введем определение пространственной оболочки.

Пространственной оболочкой трехмерного объекта называется минимальный прямоугольный параллелепипед, целиком содержащий внутри

себя данный объект. Аналогично можно определить двумерную и одномерную пространственные оболочки.

Метод состоит из трех основных шагов:

1. Упорядочение всех многоугольников в соответствии с их наибольшими z-координатами.
2. Разрешение всех неопределенностей, которые возникают при перекрытии z-оболочек многоугольников.
3. Преобразование каждого из многоугольников в растровую форму, производимое в порядке уменьшения их наибольшей z-координаты.

Ближайшие многоугольники преобразуются в растровую форму последними и закрывают более отдаленные многоугольники, так как изображаются поверх предыдущих. Реализация пунктов 1 и 3 достаточно очевидна. Рассмотрим подробнее пункт 2.

Пусть многоугольник P после упорядочения находится в конце списка, то есть является наиболее удаленным. Все многоугольники Q чьи оболочки перекрываются с z-оболочкой P должны проходить проверку по пяти тестам (шагам). Если на некотором шаге получен утвердительный ответ, то P сразу преобразуется в растровую форму.

Пять тестов:

1. x-Оболочки многоугольников не перекрываются, поэтому сами многоугольники тоже не перекрываются.
2. y-Оболочки многоугольников не перекрываются, поэтому сами многоугольники тоже не перекрываются.
3. P полностью расположен с той стороны от плоскости Q, которая дальше от точки зрения (этот тест дает положительный ответ как показано на рис. 36 a).
4. Q полностью расположен с той стороны от плоскости P, которая ближе к точке зрения. Этот тест дает положительный ответ как показано на рис. 36 b).
5. Проекции многоугольников на плоскости xOy, то есть на экране, не перекрываются (это определяется сравнением ребер одного многоугольника с ребрами другого).

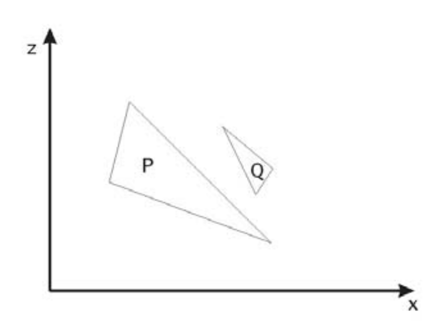
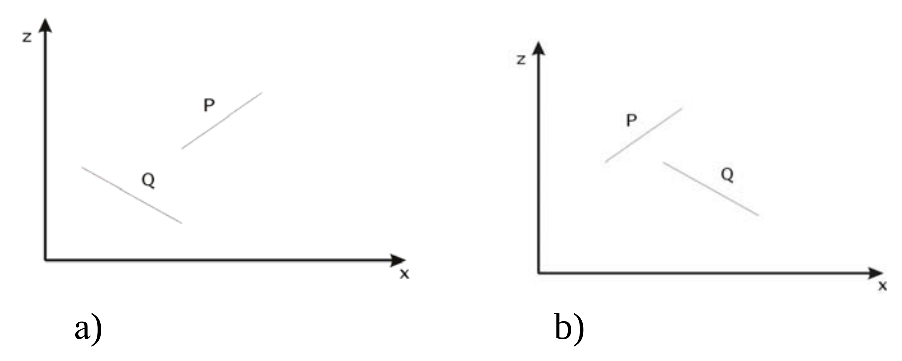


Рис. 35. z-оболочки треугольников P и Q – пересекаются.



a) b)

Рис. 36. Взаимные расположения треугольников в пространстве.

Если во всех пяти тестах получен отрицательный ответ, то P – действительно закрывает Q. Тогда меняем P и Q в списке местами. В случае, как показано на рис. 37, алгоритм зацикливается.

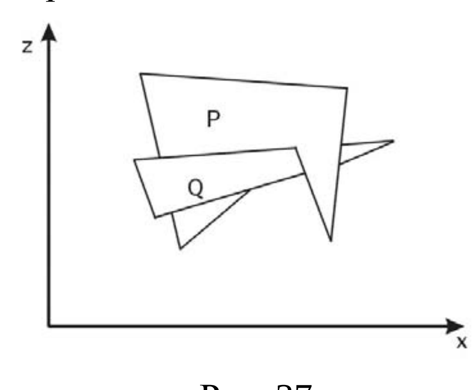


Рис. 37.

Для избежания зацикливания вводится ограничение: многоугольник, перемещенный в конец списка (т.е. помеченный), не может быть повторно перемещен. Вместо этого многоугольник P или Q разделяется плоскостью другого на два новых многоугольника. Эти два новых многоугольника

включаются в соответствующие места упорядоченного списка, и алгоритм продолжает работу.

В отличие от универсальных алгоритмов узкоспециализированный алгоритм удаления невидимых граней выпуклых тел позволяет производить вычисления гораздо быстрее. Он работает для центральной перспективной проекции. Рассмотрим работу этого алгоритма на примере как изображено на рис. 38.

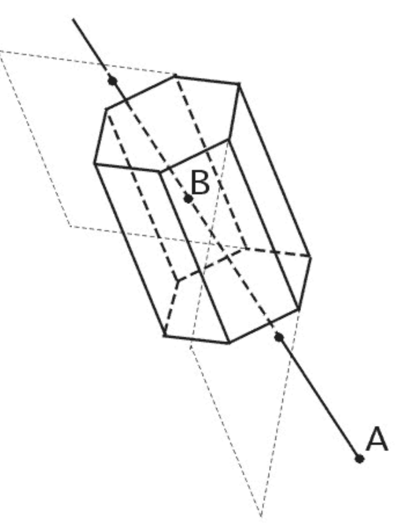


Рис. 38. Пересечения прямой AB с плоскостями граней призмы.

Пусть наблюдатель находится в точке *A*. Выберем точку *B*, которая заведомо является внутренней для выпуклой фигуры, в данном случае призмы. Выберем некоторую грань, про которую мы хотим узнать видима она из точки *A*, или не видима. Построим плоскость, в которой лежит выбранная грань. Найдем точку пересечения плоскости и прямой, которая образована отрезком *AB*. Если точка пересечения прямой и плоскости лежит внутри отрезка *AB*, то делаем вывод, что данная грань видима. Если точка пересечения находится вне отрезка *AB*, то грань не видима. В случае, когда прямая и плоскость параллельны, считаем что грань не видима.

Модели расчета освещенности граней трехмерных объектов

Основной характеристикой света в компьютерной графике является яркость. Поскольку яркость является субъективным понятием, основанным на человеческом восприятии света, то для численных расчетов применяется термин интенсивность, что соответствует яркости и является энергетической характеристикой световой волны. В расчетах интенсивность обычно

принимает значения от 0 до 1. При этом интенсивность равна нулю при полном отсутствии света, а значение 1 соответствует максимальной яркости.

В компьютерной графике для расчета освещенности граней объектов зачастую применяется трехкомпонентная цветовая модель “Красный, Зеленый, Синий”, что в английском варианте записывается RGB (Red, Green, Blue). Эта модель позволяет задавать любой цвет в виде трех компонент интенсивностей базовых цветов: красного, зеленого и синего. Интенсивность отраженного света точек пространственных объектов вычисляют отдельно для каждой их трех составляющих цветовых компонент, а затем объединяют в результирующую тройку цветов. Далее будем считать что примеры расчета интенсивностей отраженного света применяются к каждому их трех базовых цветов.

При расчете освещенности граней применяют следующие типы освещения и отражения света от поверхностей.

- Рассеянное
- Диффузное
- Зеркальное

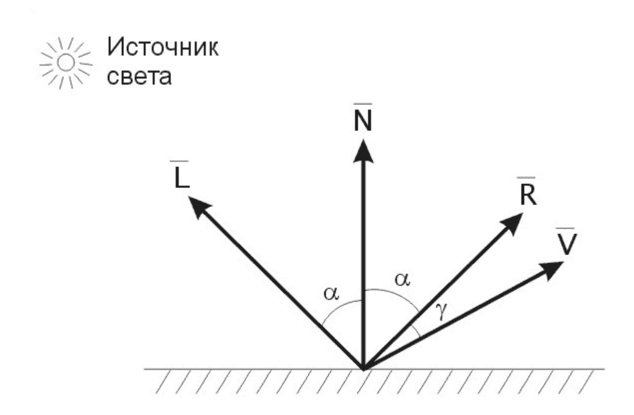


Рис. 39. Расчет интенсивности отраженного света.

Интенсивность освещения граней трехмерных объектов рассеянным светом считается постоянной в любой точке пространства. Она обусловлена множественными отражениями света от всех объектов в пространстве. При освещении трехмерного объекта рассеянным светом интенсивность отраженного света вычисляется как $I_a = I_p \cdot k_a$, где I_p- интенсивность падающего света, $k_a \in [0,1]$- коэффициент рассеянного отражения, зависит от отражающих свойств материала грани.

Для расчета интенсивности диффузного отражения света может применяться закон косинусов Ламберта: $I_d = I_p \cdot k_d \cdot Cos(\alpha)$, где α - угол падения, рассчитывается как угол между направлением на источник света и нормалью к поверхности. Пусть направление на источник света представлено единичным вектором $\overline{L}$, а $\overline{N}$ - единичный вектор нормали. Тогда $Cos\alpha = (\overline{L}, \overline{N})$ - скалярное произведение векторов. Тогда $I_d = I_p \cdot k_d \cdot (\overline{L}, \overline{N})$, где k_d - коэффициент диффузного отражения.

Вычисление зеркально отраженного света производится также с помощью различных эмпирических моделей, которые позволяют учитывать реальную шероховатость поверхностей. Например, в модели, предложенной Фонгом, интенсивность зеркально отраженного света рассчитывается в зависимости от степени отклонения от истинного значения вектора зеркально отраженного луча света. Пусть $\overline{R}$ - вектор зеркально отраженного луча света, а $\overline{V}$ - вектор, определяющий направление на наблюдателя. Тогда интенсивность зеркально отраженного света по модели Фонга рассчитывается так: $I_m = I_p \cdot k_m Cos^n \gamma$, где γ - угол между векторами $\overline{R}$ и $\overline{V}$. Константа n – может принимать значения от 1 до примерно 200, в зависимости от отражающей способности материала. Большим значениям n соответствует большая степень “гладкости” или “зеркальности” поверхности. Если векторы $\overline{R}$ и $\overline{V}$ - нормированы, то формула преобразуется к виду: $I_m = I_p \cdot k_m (\overline{R}, \overline{V})^n$.

Интенсивность отраженного света уменьшается обратно пропорционально квадрату расстояния от источника до наблюдателя. Поэтому можно записать формулу расчета интенсивности отраженного луча света для трех составляющих: рассеянного, диффузного и зеркального отражения с учетом расстояния:

$$I = I_p k_a + \frac{I_p}{R + r^2}\left(k_d \cdot (\overline{L}, \overline{N}) + k_m (\overline{R}, \overline{V})^n \right),$$

где r - расстояние от точки отражения до наблюдателя, а $R \geq 1$ - некоторая константа. Иногда, для ускорения вычислений, берут не вторую, а первую степень расстояния r.

В системах компьютерной визуализации также учитываются такие свойства материалов отражающих поверхностей как прозрачность, преломление и свечение. Степень прозрачности материала грани может описываться с помощью константы, принимающей значение от нуля до единицы, причем значение 1 соответствует полной непрозрачности материала грани. Пусть интенсивности отраженного света двух перекрывающихся

поверхностей равны I_1 и I_2. Пусть первая поверхность находится ближе к наблюдателю и является полупрозрачной с коэффициентом прозрачности α. Тогда суммарная интенсивность отраженного света может быть вычислена как взвешенное среднее: $I = I_1\alpha + I_2(1-\alpha)$.

Модели для вычисления эффектов преломления и свечения здесь не рассматриваются.

Кубические сплайны

Рассмотрим задачу проведения гладких кривых по заданным граничным точкам, или задачу интерполяции. Поскольку через две точки можно провести сколь угодно много гладких кривых, то для решения этой задачи необходимо ограничить класс функций, которые будут определять искомую кривую. Математическими сплайнами называют функции, используемые для аппроксимации кривых. Важным их свойством является простота вычислений. На практике часто используют сплайны вида полиномов третьей степени. С их помощью довольно удобно проводить кривые, которые интуитивно соответствуют человеческому субъективному понятию гладкости. Термин "сплайн" происходит от английского spline – что означает гибкую полоску стали, которую применяли чертежники для проведения плавных кривых, например, для построения обводов кораблей или самолетов.

Рассмотрим в начале сплайновую функцию для построения графика функции одной переменной. Пусть на плоскости задана последовательность точек $\{x_i, y_i\}, i=\overline{0,m}$, причем $x_0 < x_1 <,\ldots, x_{m-1} < x_m$. Определим искомую функцию $y = S(x)$, причем поставим два условия:

1) Функция должна проходить через все заданные точки: $S(x_i) = y_i$, $i=\overline{0,m}$.
2) Функция должна быть дважды непрерывно дифференцируема, то есть иметь непрерывную вторую производную на всем отрезке $[x_0, x_m]$.

На каждом из отрезков $[x_i, x_{i+1}]$, $i=\overline{0,m-1}$ будем искать нашу функцию в виде полинома третьей степени:

$$S_i(x) = \sum_{j=0}^{3} a_{ij}(x - x_i)^j .$$

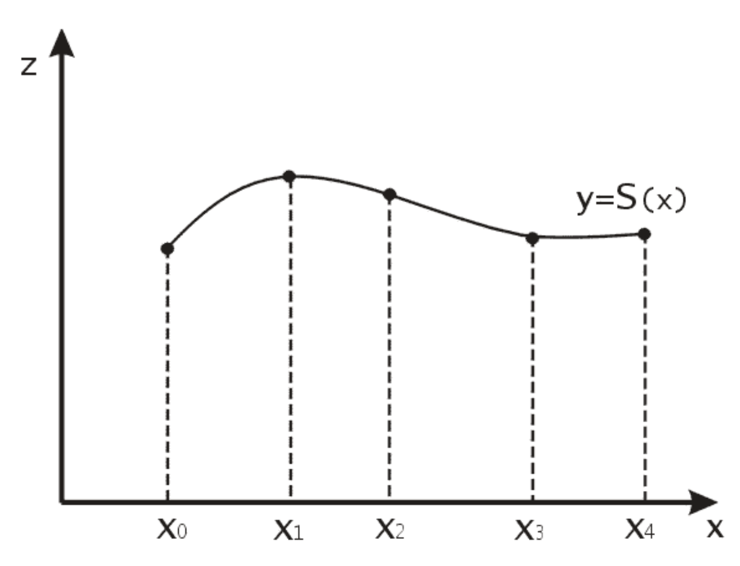


Рис. 40. Сплайновая функция.

Задача построения полинома сводится к нахождению коэффициентов a_{ij}. Поскольку для каждого из отрезков $[x_i, x_{i+1}]$ необходимо найти 4 коэффициента a_{ij}, то всего количество искомых коэффициентов будет $4m$. Для нахождения всех коэффициентов определим соответствующее количество уравнений. Первые $(m-1)$ уравнений получаем из условий совпадения значений функции во внутренних узлах $x_i, i=\overline{1, m-1}$. Следующие $2(m-1)$ уравнений получаем аналогично из условий совпадения значений первых и вторых производных во внутренних узлах. Вместе с первым условием получаем $m-1+m-1+m-1+m+1=4m-2$ уравнений. Недостающие два уравнения можно получить заданием значений первых производных в концевых точках отрезка $[x_0, x_m]$. Так могут быть заданы граничные условия.

Перейдем к более сложному случаю – заданию кривых в трехмерном пространстве. В случае функционального задания кривой $\begin{cases} y = f(x) \\ z = f(x) \end{cases}$ возможны многозначности в случае самопересечений и неудобства при значениях производных равных ∞. Ввиду этого будем искать функцию в параметрическом виде. Пусть t - независимый параметр, такой что $0 \le t \le 1$. Кубическим параметрическим сплайном назовем следующую систему уравнений:

$$\begin{cases} x(t) = a_x t^3 + b_x t^2 + c_x t + d_x \\ y(t) = a_y t^3 + b_y t^2 + c_y t + d_y \\ z(t) = a_z t^3 + b_z t^2 + c_z t + d_z \end{cases}$$

Координаты точек на кривой описываются вектором $(x(t), y(t), z(t))$, а три производные задают координаты соответствующего касательного вектора в точке. Например, для координаты x:

$$\frac{dx}{dt} = 3a_x t^2 + 2b_x t + c_x \ .$$

Одним из способов задания параметрического кубического сплайна является указание координат начальной и конечной точек, а также векторов касательных в них. Такой способ задания называется формой Эрмита. Обозначим концевые точки P_1 и P_4, а касательные векторы в них R_1 и R_4. Индексы выбраны таким образом с учетом дальнейшего изложения.

Будем решать задачу нахождения четверки коэффициентов a_x, b_x, c_x, d_x, так как для оставшихся двух уравнений коэффициенты находятся аналогично. Запишем условие для построения сплайна:

$$x(0) = P_{1x},\ x(1) = P_{4x},\ x'(0) = R_{1x},\ x'(1) = R_{4x} \qquad (*)$$

Перепишем выражение для x в векторном виде:

$$x(t) = \left[t^3, t^2, t, 1\right] \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}_x .$$

Обозначим вектор строку $T = \left[t^3, t^2, t, 1\right]$ и вектор столбец коэффициентов $C_x = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}_x$, тогда $x(t) = TC_x$.

Из (*) следует, что $x(0) = P_{1x} = [0,0,0,1]C_x$, $x(1) = P_{4x} = [1,1,1,1]C_x$. Для касательных $x'(t) = \left[3t^2, 2t, 1, 0\right]C_x, \Rightarrow$

$x'(0) = R_{1x} = [0,0,1,0]C_x$,

$x'(1) = R_{4x} = [3,2,1,0]C_x$. Отсюда получаем векторно-матричное уравнение:

$$\begin{bmatrix} P_{1x} \\ P_{4x} \\ R_{1x} \\ R_{4x} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} C_x .$$

Эта система решается относительно C_x нахождением обратной матрицы размером 4×4.

$$C_x = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_{1x} \\ P_{4x} \\ R_{1x} \\ R_{4x} \end{bmatrix} = M_h G_{hx} .$$

Здесь M_h - эрмитова матрица, G_h - геометрический вектор Эрмита. Подставим выражение C_x для нахождения $x(t)$: $x(t) = TM_h G_{hx}$. Аналогично для остальных координат: $y(t) = TM_h G_{hy}$, $z(t) = TM_h G_{hz}$.

Выпишем в явном виде формулы для вычисления координат точек сплайна. Так как $TM_h = \left[\left(2t^3 - 3t^2 + 1\right), \left(-2t^3 + 3t^2\right), \left(t^3 - 2t^2 + t\right), \left(t^3 - t^2\right)\right]$, то умножая справа на G_{hx}, получаем:

$$x(t) = TM_h G_{hx} =$$
$$= P_{1x}\left(2t^3 - 3t^2 + 1\right) + P_{4x}\left(-2t^3 + 3t^2\right) + R_{1x}\left(t^3 - 2t^2 + t\right) + R_{4x}\left(t^3 - t^2\right).$$

Четыре функции в скобках называются функциями сопряжения.

Форму кривой, заданной в форме Эрмита, легко изменять если учитывать, что направление вектора касательной задает начальное направление, а модуль вектора касательной задает степень вытянутости кривой в направлении этого вектора, как показано на рис. 41.

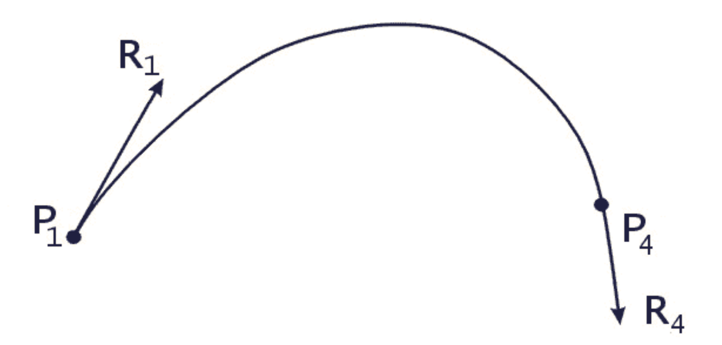


Рис. 41. Параметрический сплайн в форме Эрмита. Вытянутость кривой вправо обеспечивается тем, что $|R_1| > |R_4|$.

Рассмотрим форму Безье, которая отличается от формы Эрмита способом задания граничных условий, а именно, вместо векторов R_1 и R_4 вводятся точки (и соответствующие им радиус векторы) P_2 и P_3, как показано на рис.42, такие что выполняются условия: $P'(0) = R_1 = 3(P_2 - P_1)$ и $P'(1) = R_4 = 3(P_4 - P_3)$.

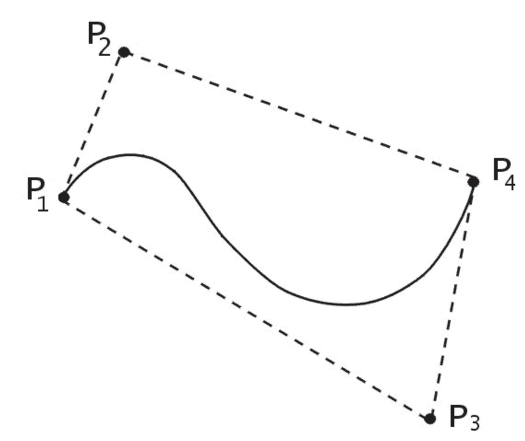


Рис. 42. Параметрический сплайн в форме Безье.

Переход от формы Эрмита к форме Безье осуществляется преобразованием:

$$G_h = \begin{bmatrix} P_1 \\ P_4 \\ R_1 \\ R_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -3 & 3 & 0 & 0 \\ 0 & 0 & -3 & 3 \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix} = M_{hb} G_b \ , \qquad (*)$$

где G_b - геометрический вектор Безье. Подставляя это в выражение для $x(t)$, получаем

$$x(t) = TM_h G_{hx} = TM_h M_{hb} G_{bx} = \left(1 - t^3\right) P_1 + 3t(t-1)^2 P_2 + 3t^2 (1-t) P_3 + t^3 P_4 \ .$$

Полезным свойством сплайнов в форме Безье является то что кривая всегда лежит внутри выпуклой оболочки, образованной четырехугольником $(P_1 P_2 P_3 P_4)$. Это свойство можно доказать, пользуясь тем, что в выражении (*) коэффициенты принимают значения от 0 до 1 и их сумма равна единице.

Заметим, что матрица вида

$$M_h M_{hb} = M_b = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$ - называется матрицей Безье.

Список литературы

1. Ньюмен, Спрулл, Основы интерактивной машинной графики, М. Мир, 1976.
2. Энджел Й. Практическое введение в машинную графику, Радио и Связь, 1984.
3. А. Вэн-Дэм, Дж. Фоли, Основы интерактивной машинной графики, т.1-2, М. Мир, 1985.
4. Е.В. Жикин, А.В.Боресков, Компьютерная графика. Динамика, реалистические ихображения, М., Диалог-МИФИ, 1995, 1997.
5. Л. Аммерал, Машинная графика на языке С, в 4-х томах, изд-во Сол. Систем, 1992.
6. Компьютер обретает разум. Пер. с англ. Под ред. В.Л.Стефанюка, М. Мир, 1990.
7. Роджерс, алгоритмические основы машинной графики. М. Мир, 1989.
8. Грайс, Графические средства персональных компьютеров, М., Мир, 1980.
9. Роджерс, Адамс, Математические основы машинной графики, М. Машиностроение, 1985.
10. Гилой, Интерактивная машинная графика, М., Мир, 1981.
11. Ф. Препарата, М. Шеймос, Вычислительная геометрия: Введение, М. Мир, 1989.
12. А.Фокс, М. Пратт, Вычислительная геометрия, М., Мир, 1982.
13. А.Б.Боресков, Е.В.Шикина, Г.Е.Шикина, Компьютерная графика: первое знакомство, Под ред. Е.В.Шикина, М., Финансы и статистика, 1996.
14. А.В.Фролов, Г.В.Фролов, Графический интерфейс GDI в MS WINDOWS, Москва, Изд-во Диалог-МИФИ, 1994.
15. Майкл Ласло, Вычислительная геометрия и компьютерная графика на С++, Москва, Бином, 1997.
16. Ю.Тихомиров, Программирование трехмерной графики, С.-Пб.: БХВ-Санкт-Петербург,1999.
17. А.Хонич, Как самому создать трехмерную игру. М.:МИКРОАРТ, 1996.
18. М.Маров, 3D Studio MAX 2.5: справочник – СПб: «Питер», 1999. – 672 с.
19. А.Ла Мот, Д.Ратклифф и др. Секреты программирования игр/ Перев с англ. – СПб: Питер, 1995. – 720 с.
20. Н. Томпсон, Секреты программирования трехмерной графики для Windows 95. Перев с англ. – СПб: Питер, 1997. – 352 с.